OFFICE OF NAVAL RESEARCH

FINAL REPORT

FOR

Grant No.: N00014-91-J-1852
R & T Project: 4148503-03

VLSI for High-Speed Digital Signal Processing

1 July 1991 through 30 September 1994

Principal Investigator: Alan N. Willson, Jr.

7400 Boelter Hall
University of California, Los Angeles
405 Hilgard Avenue
Los Angeles, CA 90024-1600
(310) 825-7400
e-mail: willson@ee.ucla.edu

Scientific Officer: Dr. Clifford Lau

VLSI for High-Speed Digital Signal Processing

The research supported by this ONR grant has investigated modern high-speed signal processing system design. It has encompassed a complete spectrum of activities, starting with the discovery of new signal processing algorithms, and continuing through the development of the most appropriate methods for their realization, including, in particular, the design, layout and fabrication of integrated circuits.

The primary project for this grant has been the design and implementation of a new type of programmable general purpose digital filter IC. It employs multiple processing units on a single integrated circuit. The multiple processors operate in parallel and communicate with one another through on-chip dual-access storage register blocks, thereby incurring no operating speed penalties as would result if it were necessary to read and write to off-chip RAM. The system's topology has the processors arranged in a ring, with locally-shared register blocks between each adjacent pair of processors. Our prototype IC has five processors, and it is capable of realizing a rich variety of filter structures that operate at the maximum instruction execution rate possible for any custom parallel implementation.

A circuit board using four of our ring processor chips was also designed and built. It demonstrates the IC's capability to perform real-time video processing and high-speed one-dimensional processing of data. It resides in an IBM PC computer and is accessed through the PC bus. A custom software package was also written that facilitates the programming of the chips and the configuration of the circuit board for each of its several operating modes.

In addition to the design of the above-mentioned ring-of-processors IC we have also developed a task partitioner, which is a computer program that automatically writes programs for our ring of parallel processors. It accepts an arbitrary filter's description in net-list form and calculates the theoretical optimum sampling period for the filter's structure on a multiprocessor system with P processors. (In our case we of course set P=5.) Our algorithm detects all multiple-input adders in the desired filter structure and provides the user the option of searching for the optimum adder sequence to minimize the filter's sampling period. It then determines the optimum time schedule, and optimally distributes the computations over the processors in the ring. The program's output is a set of programs for the parallel processors which causes them to implement the desired filter structure. We have found the task scheduler capable of implementing all practical examples of digital filter structures at the optimum sampling period.

Another project supported by this grant has concerned the design, layout, and fabrication of a programmable digital signal processor using switchable unit-delays for optimal coefficient allocation in the implementation of FIR filters. This architecture enables very high-speed processing (Our prototype IC proved capable of implementing FIR filters having data rates of 180 MHz.) while avoiding the severe hardware inefficiency that would result from straightforward programmable tap implementation such as the types that had been reported previously. The switchable unit-delay not only allows the programming of the number of filter taps and the specific filter-tap coefficient values, it provides the capability for programming the *optimal allocation of hardware resources* to each filter tap. We fabricated a prototype chip capable of realizing a broad spectrum of linear-phase FIR filters employing up to 32 taps. It was designed using Mentor

Graphics GDT VLSI CAD tools, and we wrote a silicon compiler in the Genie language to assemble the chip with parameterized word length and number of taps.

Another project that was supported by this ONR grant concerned an improvement to the Powell and Chau linear phase IIR filters. In this work we developed a technique using Jacobian elliptic functions which, by removing a previous method's double-zero constraint, yields improved designs of linear phase IIR filters.

Other research carried out under the auspices of this grant dealt with the design of two-channel perfect-reconstruction linear-phase FIR filter banks. Two new approaches were developed for the design of such filter banks: the first, formulating the problem as a quadratic programming problem with linear constraints, and the second, as one with nonlinear constraints. We also developed an optimization technique for the design of *multiplierless* two-channel linear-phase FIR filter banks employing canonic signed-digit (CSD) code using the new structures, and another technique was developed for lattice-structure perfect-reconstruction filter banks with powers-of-two coefficients.

One further project supported by this ONR grant is worth explicit mention. A silicon compiler for Recursive Running Sum (RRS) and Simple Symmetric Sharpening (SSS) digital filter structures was written and used to produce a prototype IC. These structures have been shown by Adams and Willson to offer significant advantages in prefilter-equalizer type implementations of FIR filters. The prototype IC was designed to achieve a throughput rate of 175 MHz in 1.2-μm CMOS.

The following students earned Master's degrees with theses or projects supported by this ONR grant: M. C.P. Chen, M. L. Coulter, H. T. Hung, M. C.Kennedy, K-Y. Khoo, A. Y. Kwentus, L. T-P. Ying. Four of these students are presently employed in industry and three (Chen, Khoo and Kwentus) are currently continuing their UCLA studies toward Ph.D. degrees. The research has been documented in *seven* journal publications (and in two additional journal publication currently under review) and 13 conference papers, itemized in the annual reports. One invited lecture via the UCLA/SNU telelink was presented to Seoul National University, and one patent application was filed based on the grant's research.

In the following pages we give a thorough description of the Ring-of-Processors IC, the grant's major project. We also describe in some detail the programmable digital signal processor using switchable unit-delays for optimal coefficient allocation in the implementation of FIR filters, the project for which a patent application was filed. Reprints of journal papers discussing the task partitioner research, the improvement to the Powell and Chau linear phase IIR filters, and the perfect-reconstruction linear-phase FIR filter banks are incorporated into this report, as is the brief description of the project on the implementation of high-speed programmable digital FIR prefilters, which is the text of the paper awarded third prize in the recent national IC design competition. (This competition was sponsored by Mentor Graphics, Electronic Design, Hewlett Packard, Sun Microsystems, and Texas Instruments.) Separate copies of all journal papers and conference papers have been submitted, and copies of all UCLA master's theses are available from the principal investigator upon request.

VLSI for High-Speed Digital Signal Processing
1 July 1991 - 30 September 1994

Published Papers in Refereed Journals

B-R. Horng, H. Samueli, and A. N. Willson, Jr., "The Design of Low-Complexity Linear-Phase FIR Filter Banks Using Powers-of-Two Coefficients with an Application to Subband Image Coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 1, pp. 318-324, December 1991. (ONR)

B-R. Horng and A. N. Willson, Jr., "Lagrange Multiplier Approaches to the Design of Two-Channel Perfect-Reconstruction Linear-Phase FIR Filter Banks," *IEEE Trans. on Signal Processing*, vol. 40, pp. 364-374, February 1992. (ONR, NSF)

A. Y. Kwentus, M. J. Werter, and A. N. Willson, Jr., "A Programmable Digital Filter IC Employing Multiple Processors on a Single Chip," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 2, pp. 231-244, June 1992. (ONR, MICRO)

J. L. White and A. N. Willson, Jr., "On the Equivalence of Spatial and Temporal Stability for Translation Invariant Linear Resistive Networks," *IEEE Trans. on Circuits and Systems - I*, vol. 39, pp. 734-743, September 1992. (ONR, NSF)

B-R. Horng, H. Samueli, and A. N. Willson, Jr., "The Design of Two-Channel Lattice-Structure Perfect-Reconstruction Filter Banks Using Powers-of-Two Coefficients," *IEEE Trans. on Circuits and Systems - I*, vol. 40, pp. 497-499, July 1993. (ONR)

A. Y. Kwentus, H-T. Hung, and A. N. Willson, Jr., "An Architecture for High-Performance / Small-Area Multipliers for Use in Digital Filtering Applications," *IEEE Journal of Solid-State Circuits*, vol. 29, pp. 117-121, February 1994. (ONR)

M. J. Werter and A. N. Willson, Jr., "Automated Programming of Digital Filters for Parallel Processing Implementation," *IEEE Trans. on Circuits and Systems - II*, vol. 41, pp. 285-294, April 1994. (ONR)

Papers Submitted to Refereed Journals (and not yet published)

K-Y. Khoo, A. Y. Kwentus, and A. N. Willson, Jr., "An Efficient 180 MHz Programmable FIR Digital Filter," submitted to *IEEE Journal of Solid-State Circuits*, 1994. (ONR, MICRO)

A. N. Willson, Jr. and H. J. Orchard, "An Improvement to the Powell and Chau Linear Phase IIR Filters," to appear in the October 1994 issue of *IEEE Trans. on Signal Processing*. (ONR, NSF)

Books or Chapters Published

Chapter: A Ring-Structured Topology of Programmable Digital Filter Processors on a Single Chip, pp. 195-204, in *VLSI Signal Processing, V.* IEEE, edited by K. Yao and R. Jain, 1992.

Patents Filed

A Programmable Digital Signal Processor Using Switchable Unit-Delays for Optimal Hardware Allocation (Patent filed, April 1993)

Invited Presentations at Workshops or Professional Society Meetings

A. N. Willson, Jr., "A Programmable Digital Filter Integrated Circuit Employing Multiple Processors," invited lecture via UCLA / SNU telelink to Electrical Engineering Seminar at Seoul National University, April 29, 1992.

A. N. Willson, Jr., "Transistor Network Operating Point Theory." Keynote lecture, Nonlinear Circuit Analysis and Simulation Workshop, AT&T Bell Laboratories, Murray Hill, NJ, August 8-9, 1994.

Contributed Presentations at Workshops or Professional Society Meetings

C-Y. Yao, and A. N. Willson, Jr., "One-Neuron Circuitry for Carry Generation in a 4-bit Adder," *Proc. of the 1992 International Joint Conference on Neural Networks*, sponsored by IEEE and the International Neural Network Society, Baltimore, MD, June 7-11, 1992, pp. III.179-III.184. (ONR)

A. Y. Kwentus, M. J. Werter and A. N. Willson, Jr., "A Ring-Structured Topology of Programmable Digital Filter Processors on a Single Chip," *Proc. of the 1992 URSI International Symposium on Signals, Systems, and Electronics*, sponsored by the International Union of Radio Science, Paris, September 1-4, 1992, pp. 278-281. (ONR, MICRO)

A. Y. Kwentus, M. J. Werter and A. N. Willson, Jr., "A Ring-Structured Topology of Programmable Digital Filter Processors on a Single Chip," *Proc. of the 1992 IEEE International Workshop on VLSI Signal Processing*, Napa Valley, CA, October 28-30, 1992, pp. 195-204. (ONR, MICRO)

K-Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "An Efficient 175 MHz Programmable FIR Digital Filter," *Proc. of the 1993 IEEE International Symposium on Circuits and Systems*, Chicago, IL, May 3-6, 1993, pp. 72-75. (ONR, NSF)

A. Y. Kwentus, H-T. Hung, and A. N. Willson, Jr., "High-Performance / Small-Area Multipliers for use in Digital Filtering Applications," *Proc. of the 36th Midwest Symposium on Circuits and Systems*, Detroit, MI, August 16-18, 1993, pp. 1493-1496. (ONR, MICRO)

A. Y. Kwentus, M. J. Werter, and A. N. Willson, Jr., "A Programmable Digital Filter IC Employing Multiple Processors on a Single Chip," *Proc. of the International Conference on Signal Processing Applications & Technology '93*, Santa Clara, CA, September 28-October 1, 1993, pp. 42-51. (ONR, MICRO)

K-Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "An Efficient 180 MHz Programmable FIR Digital Filter," *Proc. of the International Conference on Signal Processing Applications & Technology '93*, Santa Clara, CA, September 28-October 1, 1993, pp. 53-59. (ONR, NSF)

A. Y. Kwentus, M. J. Werter and A. N. Willson, Jr., "A Programmable Digital Filter IC Employing Multiple Processors on a Single Chip," *Proc. DSPx*, Santa Clara, CA, Oct. 5-7, 1993, pp. 299-308. (ONR, MICRO)

K-Y. Khoo, A. Y. Kwentus and A. N. Willson, Jr., "An Efficient 180 MHz Programmable FIR Digital Filter," *Proc. DSPx*, Santa Clara, CA, Oct. 5-7, 1993, pp. 309-316. (ONR, NSF)

A. N. Willson, Jr. and H. J. Orchard, "An Improvement to the Powell and Chau Linear Phase IIR Filters," *Proc. of ICASP '94*, Adelaide, Australia, April 19-22, 1994, pp. III.573-III.576. (ONR, NSF, MICRO)

K-Y. Khoo and A. N. Willson, Jr., "Low Power CMOS Clock Buffer," *Proc. of ISCAS '94*, London, May 30-June 2, 1994, pp. 355-358. (ONR, NSF)

P. Saghizadeh and A. N. Willson, Jr., "A New Approach to the Design of Three-Channel Perfect-Reconstruction Linear-Phase FIR Filter Banks," *Proc. of ISCAS '94*, London, May 30-June 2, 1994, pp. 157-160. (ONR, NSF)

P. Saghizadeh and A. N. Willson, Jr., "Using Unconstrained Optimization in the Design of Two-Channel Perfect-Reconstruction Linear-Phase FIR Filter Banks," *Proc. of the 37th Midwest Symposium on Circuits and Systems*, Lafayette, LA, August 15-17, 1994. (ONR, NSF)

Honors / Awards / Prizes

A. N. Willson, Jr., and M. M. Green: W. R. G. Baker Prize Award, IEEE.

A. N. Willson, Jr., and M. M. Green: 1994 Guillemin-Cauer Award, IEEE Circuits and Systems Society.

K-Y. Khoo: 1994 UCLA School of Engineering, Outstanding M.S. Student Award.

Linda T-P. Ying: 3rd Place Prize Award ($2,000) in 1994 Student VLSI Design Contest (a national competition sponsored by Mentor Graphics, Electronic Design, Hewlett Packard, Sun Microsystems, and Texas Instruments) for "High-Speed Programmable FIR Prefilter Implementation." This was the IC resulting from Ms. Ying's M.S. Thesis project, which was supported by this ONR grant.

Ring Project - Final Report

I. System Description.

A new programmable general-purpose digital filter IC is described that employs multiple processing units on a single chip. The multiple processors operate in parallel and communicate with one another through on-chip dual-access storage register blocks. The topology of the digital filter chip has the processors arranged as a ring with the locally shared register blocks between each adjacent pair of processors, as shown in Figure 1. Each processor has its own coefficient and program memory, program decode logic, and ALU with a hardware multiplier. As is shown in [1], this ring of processors is capable of realizing a rich variety of filter structures operating at the maximum possible instruction execution rate, i.e., requiring the minimum number of program steps per data sample that can possibly be achieved for any custom parallel-processing implementation.

Two digital filter processor ICs have been designed. The first contains a single processor with two adjacent dual-port register blocks while the second contains the complete ring of five processors with five dual-port register blocks. The first IC was intended as a test vehicle to verify the operation of all the major blocks before fabricating the full five-processor ring. The two ICs are pin-for-pin compatible, making the ring IC a "drop-in" replacement for the single-processor IC on test boards.

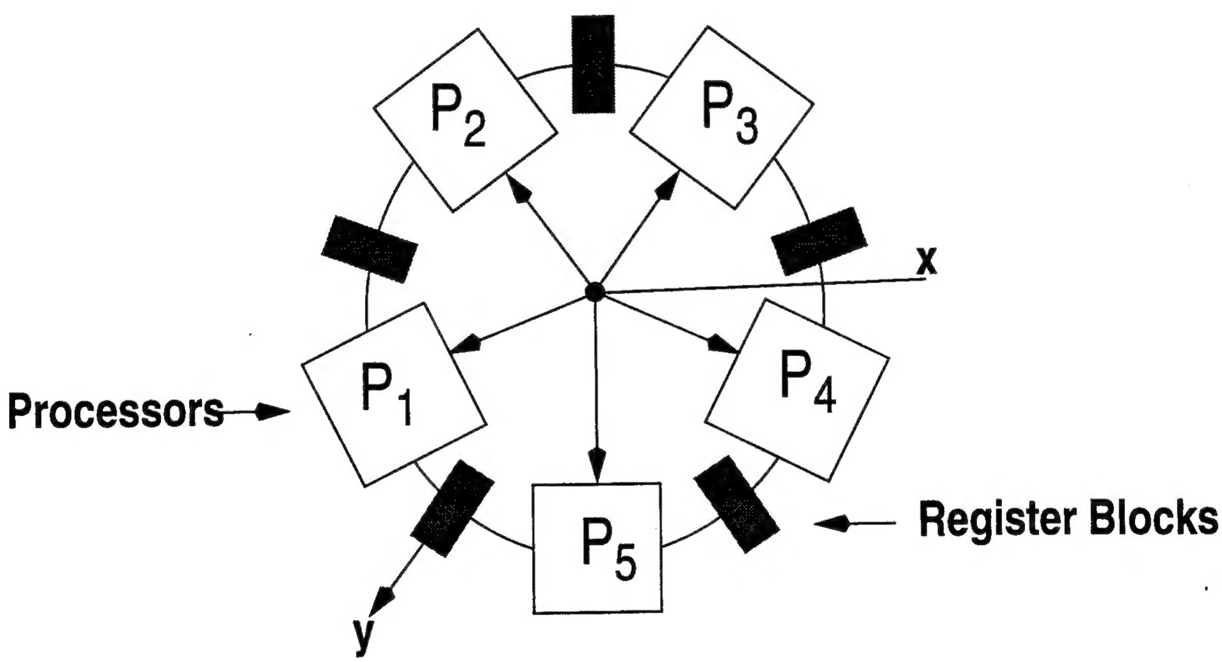


Figure 1 - Ring-structured processor topology.

A. Single-Processor IC

The single-processor IC contains a processor (consisting of a coefficient RAM, a program RAM, program decoding logic, and an ALU with a hardware multiplier) as well as clock generation

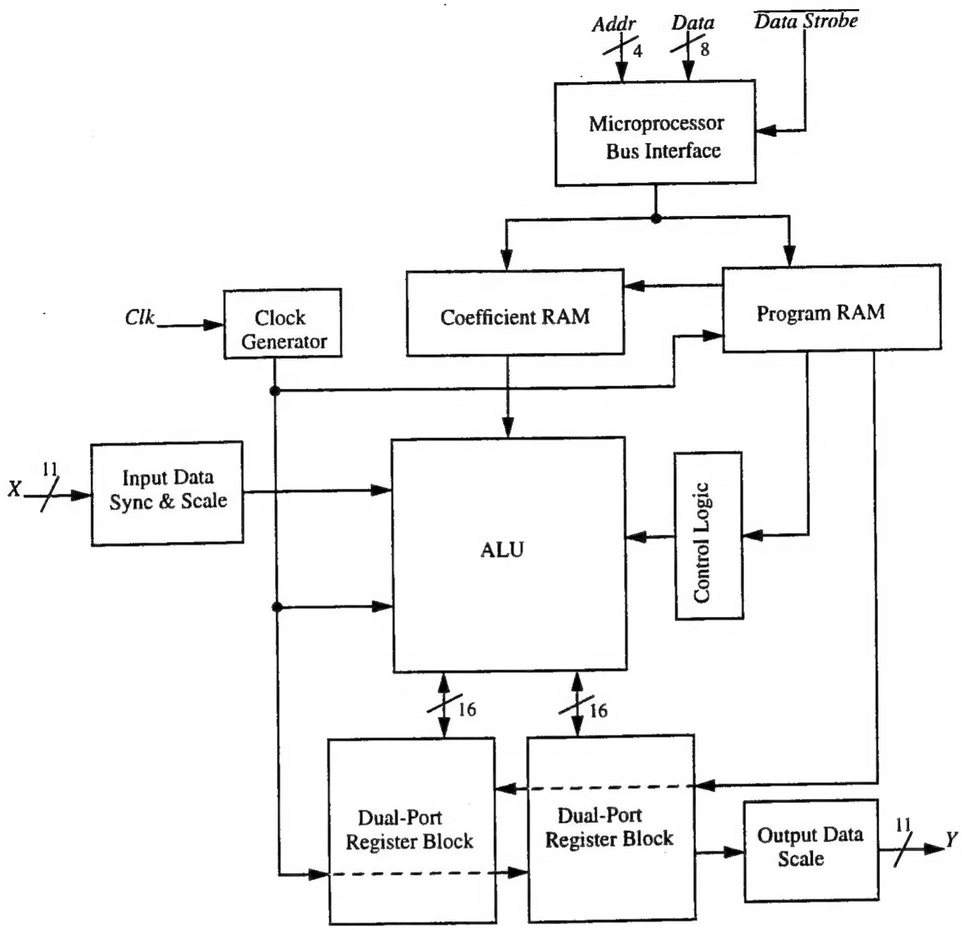


Figure 2 - Single-processor IC block diagram.

circuitry, input and output data synchronization and scaling circuitry, microprocessor bus interface circuitry, and two dual-port register blocks. It is in essence a 1-processor "ring". A block diagram of this IC is shown in Figure 2.

The IC provides for 11-bit input and output data with 16-bit internal data. 12-bit filter coefficients are stored in an internal coefficient RAM. It also includes an 8-bit microprocessor bus interface for loading programs and coefficients. The IC contains 24,723 transistors in an area of 14.8 mm^2 (including pads) and was fabricated through MOSIS in a 1.2-µm CMOS N-well process. Testing results show >50 MHz operation with a 5V supply voltage and 25 MHz operation with a 3V supply voltage. The IC was designed using the *magic* CAD tools in a scalable CMOS technology. Figure 3 shows the chip micrograph.

B. Ring-Processor IC

The ring-processor IC contains five processors and five register blocks interconnected as shown in Figure 1. Each processor consists of a coefficient RAM, a program RAM, program decoding logic, and an ALU with a hardware multiplier. The IC also contains clock generation circuitry, input and output data synchronization and scaling circuitry, and microprocessor bus interface circuitry. It is pin-for-pin compatible with the single-processor IC.

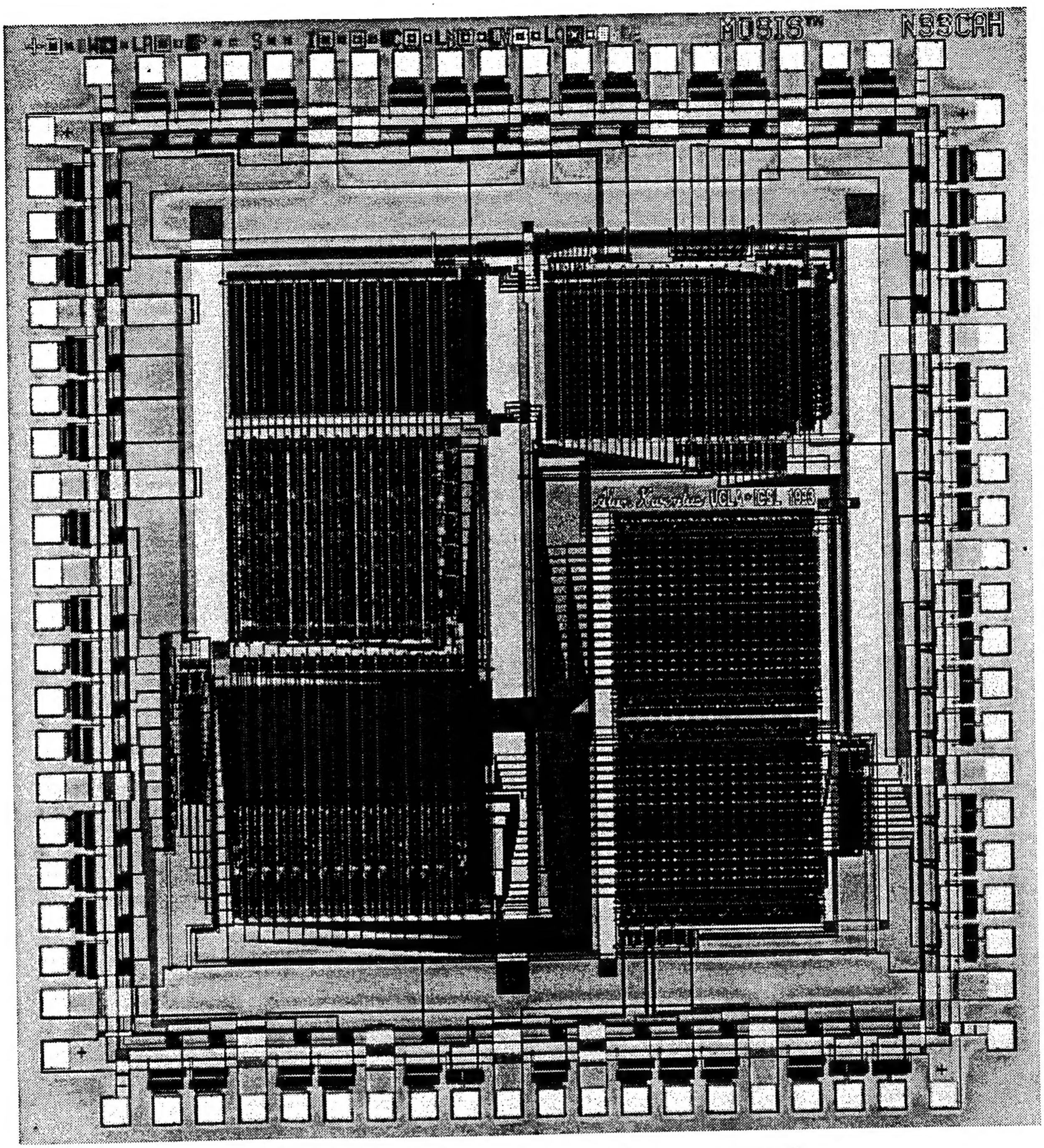


Figure 3 - Single-processor IC chip micrograph.

The core of the Ring-Processor chip has been completed and is currently undergoing extensive simulation before fabrication. The IC will be fabricated through MOSIS in a 1.2-µm CMOS N-well process (same as the single-processor IC). The IC core contains 96,378 transistors in an area of 43.8 mm^2. SPICE simulations indicate that the IC should operate at data rates > 50MHz with a 5V supply voltage. Figure 4 shows a plot of the core layout.

<u>NOTE:</u> Throughout this documentation, signals that are input or output pins appear in *italic* type while signals internal to the IC appear in regular type.

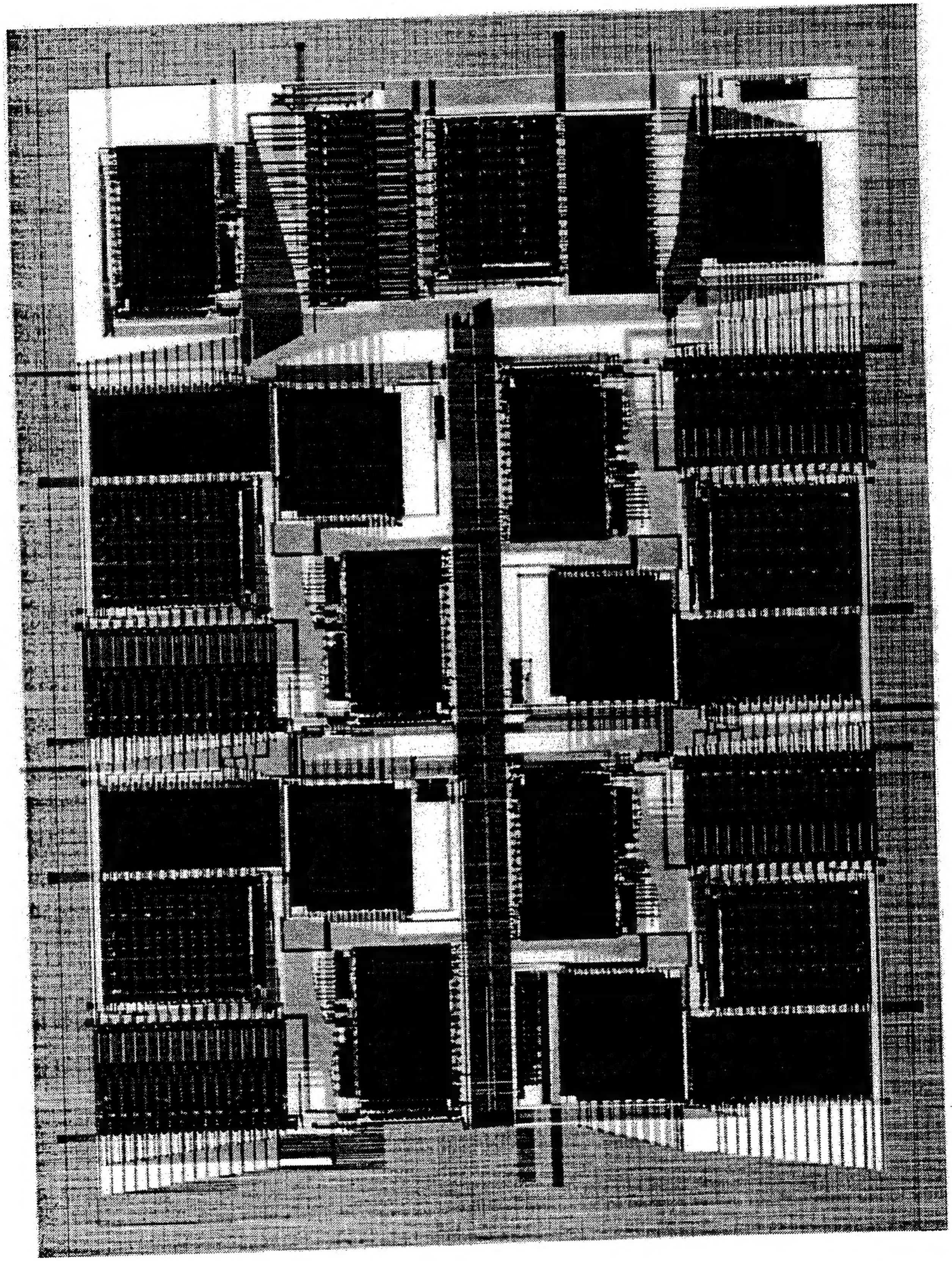


Figure 4 - Ring-processor IC core layout.

II. IC Pinout

The Single-Processor IC is packaged in an 84-pin PGA package. The Ring-Processor IC will also be packaged in an 84-pin PGA package with an identical pinout to that of the Single-Processor IC. The pinout and I/O signals are given below in Figure 5 and Table 1.

X

	L	K	J	H	G	F	E	D	C	B	A	
1	21	19	18	16	13	12	9	6	4	3	84	1
2	24	22	20	17	14	7	8	5	2	1	82	2
3	25	23			15	11	10			83	81	3
4	27	26								80	79	4
5	30	29	31						75	77	76	5
6	33	28	32						74	73	78	6
7	34	35	36						70	71	72	7
8	37	38								68	69	8
9	39	41			49	53	54			65	67	9
10	40	43	44	47	50	52	56	59	62	64	66	10
11	42	45	46	48	51	57	55	58	60	61	63	11
	L	K	J	H	G	F	E	D	C	B	A	

Figure 5 - 84-pin PGA.

Table 1: Ring-Processor and Single-Processor IC Pinout

Pin	Signal	Pin	Signal	Pin	Signal
A1	*prog*	C11	$\overline{\textit{hold\_clk}}$	J2	*data[6]*
A2	*pmux[0]*	D1	*addr[1]*	J5	*X[10]*
A3	$\overline{\textit{reset}}$	D2	*GND*	J6	*VDD*
A4	*Y_clk*	D10	*VDD*	J7	*X[6]*
A5	*Y[8]*	D11	*clk_bypass*	J10	*GND*
A6	*Y[10]*	E1	*addr[3]*	J11	*X[0]*
A7	*Y[5]*	E2	*VDD*	K1	*data[7]*
A8	*Y[3]*	E3	$\overline{\textit{data\_strobe}}$	K2	*VDD*
A9	*Y[1]*	E9	*clk*	K3	*data[5]*
A10	*Y[0]*	E10	*phi2_in*	K4	*GND*

Table 1: Ring-Processor and Single-Processor IC Pinout

Pin	Signal	Pin	Signal	Pin	Signal
A11	VDD	E11	GND	K5	data[0]
B1	load_sync	F1	proc[2]	K6	data[1]
B2	VDD	F2	addr[2]	K7	X[7]
B3	pmux[1]	F3	GND	K8	X[5]
B4	GND	F9	VDD	K9	X[2]
B5	Y[9]	F10	out_clk	K10	VDD
B6	Y[6]	F11	phi1_in	K11	X[1]
B7	Y[4]	G1	proc[1]	L1	GND
B8	Y[2]	G2	proc[0]	L2	data[4]
B9	VDD	G3	VDD	L3	data[3]
B10	GND	G9	ext_inclk	L4	data[2]
B11	phi1_out	G10	GND	L5	GND
C1	addr[0]	G11	scale	L6	X[9]
C2	GND	H1	GND	L7	X[8]
C5	VDD	H2	$\overline{chip\_select}$	L8	GND
C6	Y[7]	H10	VDD	L9	X[4]
C7	GND	H11	ext_inclk_byp	L10	X[3]
C10	phi2_out	J1	coeff	L11	VDD

III. I/O Signal Description.

Following is a list of the input and output pins and their functions.

A. Input Data.

X[10:0] input

These 11 pins provide the X input data to the ring-of-processors.

ext_inclk input

This input is used to externally clock the X-input data into the chip (see Section V.) It can be bypassed using *ext_inclk_byp*. When used, the input data is clocked into the chip on the rising edge of *ext_inclk*.

ext_inclk_byp input

This active high input is used to bypass the external input clock for the X-input data (see Section V.) When this pin is set high, the external input clock *ext_inclk* is bypassed. When this pin is set low, the external input clock *ext_inclk* is used to clock the X-input data into the chip.

B. Output Data.

Y[10:0] output

These 11 pins provide the Y data output from the ring-of-processors.

Y_clk output

This is the output data clock. An active-high pulse of width equal to one phi2 pulse width will be output when the output data *Y* changes.

C. System Clocking (see Section IV.)

clk input

This input is the system clock input. All internal timing is derived from this input clock, unless bypassed using the *clk_bypass* input. Two-phase non-overlapping clocks are generated internally from this input. Each processor executes one instruction for every cycle of *clk*.

phi1_in input

This input is used in conjunction with *phi2_in* and *clk_bypass* to provide external two-phase non-overlapping clocks to the chip.

phi2_in input

This input is used in conjunction with *phi1_in* and *clk_bypass* to provide external two-phase non-overlapping clocks to the chip.

clk_bypass input

This active high input is used to bypass the internal two-phase non-overlapping clock generator. When this pin is set low, the *clk* input is used to generate internal two-phase non-overlapping clocks (phi1 and phi2). When this pin is set high, the clock generator is bypassed and the *phi1_in* and *phi2_in* pins are used to provide the two-phase non-overlapping clocks used internally.

$\overline{\textit{hold\_clk}}$ input

This active low input is used to turn off all the internal clocks when loading programs. It can also be used to conserve power when the chip is in a standby mode. When this pin is set low, all internal clocks are turned off. When this pin is set high, all internal clocks are turned on for normal operation. Note: this pin must be set low (i.e., all internal clocks off) when loading programs into the program memory.

out_clk output

This output is a buffered copy of the system input clock *clk*. It is primarily intended for diagnostic purposes.

phi1_out output

This output is a buffered copy of the internal phi1 clock.

phi2_out output

This output is a buffered copy of the internal phi2 clock.

D. System Programming (see Sections VII. and VIII.)

data[7:0] input

These 8 bits are the program data bus used to load program and coefficient data for the internal processors.

addr[3:0] input

These 4 bits are used to select which program address or coefficient address receives the information on the program data bus. These bits are also used to load each processor's internal reset address register.

$\overline{\textit{data\_strobe}}$ input

This active low input is used to strobe the data on the program data bus into the chip.

$\overline{\textit{chip\_select}}$ input

This active low input is used to select the chip for programming information. This input does not affect the normal operation of the chip.

proc[2:0] input

These 3 input pins are used to select which processor receives the programming information on the program data bus.

prog input

This active high input is used to select the mode of operation in which programs are loaded into the chip. When *prog* is active, *coeff* and *scale* should both be inactive.

coeff input

This active high input is used to select the mode of operation in which coefficients are loaded into the chip. When *coeff* is active, *prog* and *scale* should both be inactive.

scale input

This active high input is used to select the mode of operation in which the input and output scale values are loaded into the chip. When *scale* is active, *prog* and *coeff* should both be inactive.

load_sync input

This active high input is used to synchronize the program address *addr[3:0]* with the internal clocks. When this pin is set high, the program address inputs are synchronized to the internal clocks. When this pin is set low, the program address inputs are not synchronized to the internal clocks. This pin must be set low when programming the internal processors because all internal clocks should be disabled using $\overline{hold\_clk}$ at this time. However, during normal operation, this pin must be set high to ensure proper synchronization when changing the program reset address.

pmux[1:0] input

These two inputs select which part of the program word or coefficient word the 8-bit program data bus will be written to. When loading programs onto the chip (i.e., *prog* set high), these two inputs select which 8 bits of the 32-bit program word will be written. When loading coefficients onto the chip (i.e., *coeff* set high), *pmux[1]* selects either the 1-X or 3-X coefficient and *pmux[0]* selects the LSB or MSB of the 13-bit coefficient word.

$\overline{\textbf{\textit{reset}}}$ input

This active low input is used to reset all internal processor's program counters to the address stored in their internal reset address register.

IV. System Clocking

The system operates with a two-phase non-overlapping clocking scheme. A two-phase clock generator is provided on-chip to allow for a single input clock. Alternatively, the IC can be configured to operate with two non-overlapping clock inputs. Also, a $\overline{hold\_clk}$ signal is provided to disable all internal clocks for loading programs or for reducing power during standby periods. The clock control circuitry and timing is shown in Figure 6.

Table 2: Clock Modes

$\overline{hold\_clk}$	*clk_bypass*	Clock Mode	clock input
1	0	single clock input clock generator enabled	*clk*
1	1	two clock input	*phi1_in, phi2_in*
0	X	all internal clocks disabled	--------

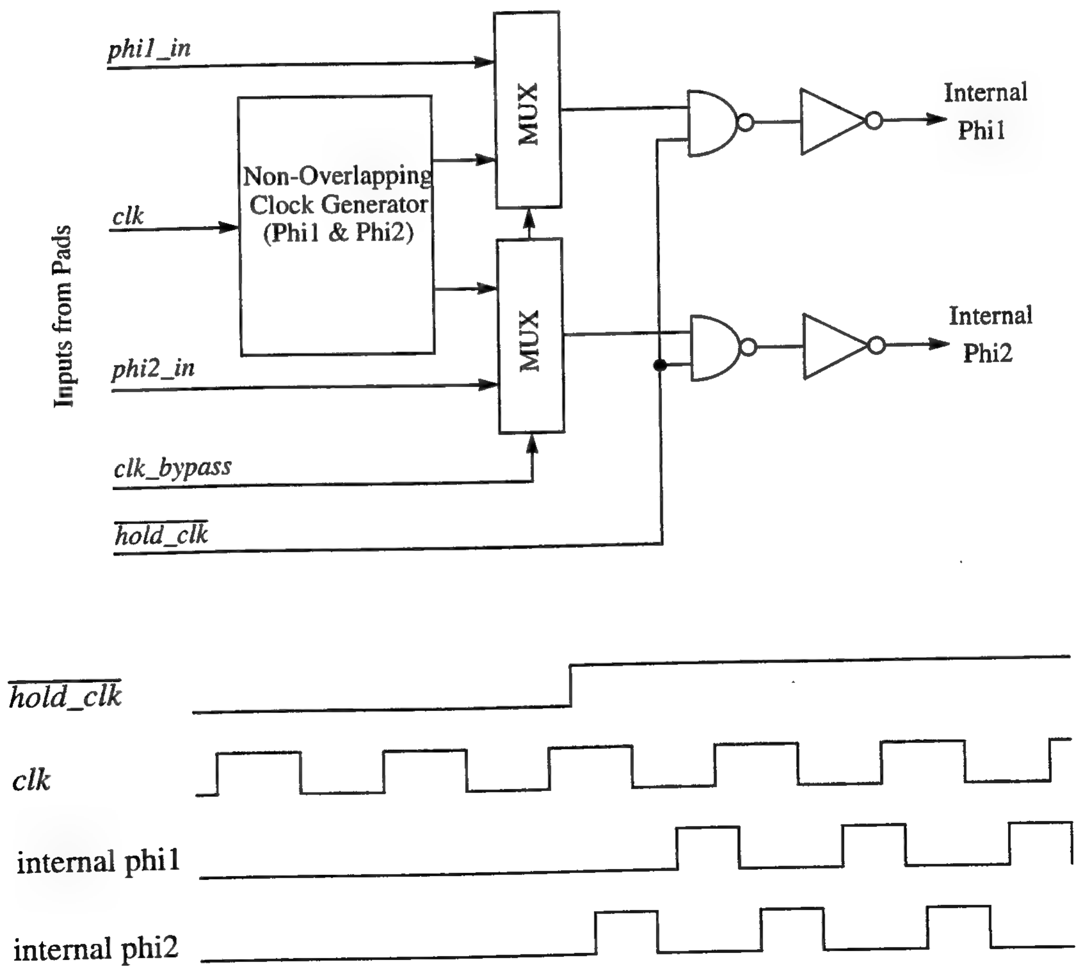


Figure 6 - Clock control circuitry and timing.

V. Input Data Synchronization and Scaling

Figure 7 shows the block diagram for the input data synchronization and scaling block of Figure 2. The input data is synchronized to the internal phi1 clock before being passed to the ALU input to insure that it will be available for input to the ALU at the same time that the other inputs are available. Additionally, an external input data clock (ext_inclk) is provided for systems that operate synchronously or for applications using a shared data bus. This external input data clock can be bypassed if not used. The input register clocked by *ext_inclk* operates as a rising edge triggered flip flop.

Table 3: Input Modes

ext_inclk_byp	Input Mode
0	external data clock (*ext_inclk*) enabled
1	external data clock (*ext_inclk*) disabled

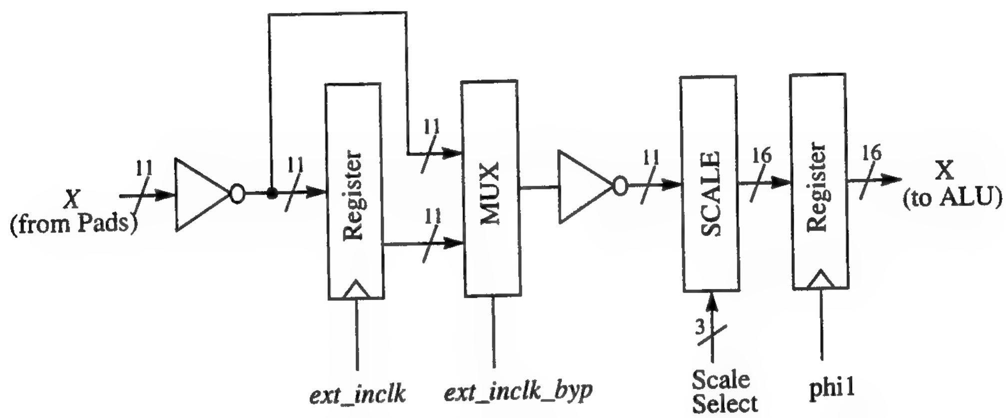


Figure 7 - Input data synchronization and scaling.

The scale block shown in Figure 7 selects which of the 16 internal bits the 11 input bits will be placed into (with sign extension when appropriate). This allows the input to be shifted by up to 5 bits to the right (i.e., scaled down by a factor of up to 32).

Table 4: Scale Select vs. ALU X-input Data

Scale	16-bit X input to the ALU															
000	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]	0	0	0	0	0
001	X[10]	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]	0	0	0	0
010	X[10]	X[10]	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]	0	0	0
011	X[10]	X[10]	X[10]	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]	0	0
100	X[10]	X[10]	X[10]	X[10]	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]	0
101	X[10]	X[10]	X[10]	X[10]	X[10]	X[10]	X[9]	X[8]	X[7]	X[6]	X[5]	X[4]	X[3]	X[2]	X[1]	X[0]

VI. Output Data Scaling

Figure 8 shows the block diagram for the output data scaling block of Figure 2. The scale block shown in the figure selects which of the internal 16 bits will be output to the 11 output pads (*Y[10:0]*). This allows for the output to be shifted to the left by up to 5 bits (i.e., scaled up by a factor of up to 32). Note that there is no overflow protection. Care must be taken to ensure that the output value (in two's complement form) does not overflow because errors could be substantial.

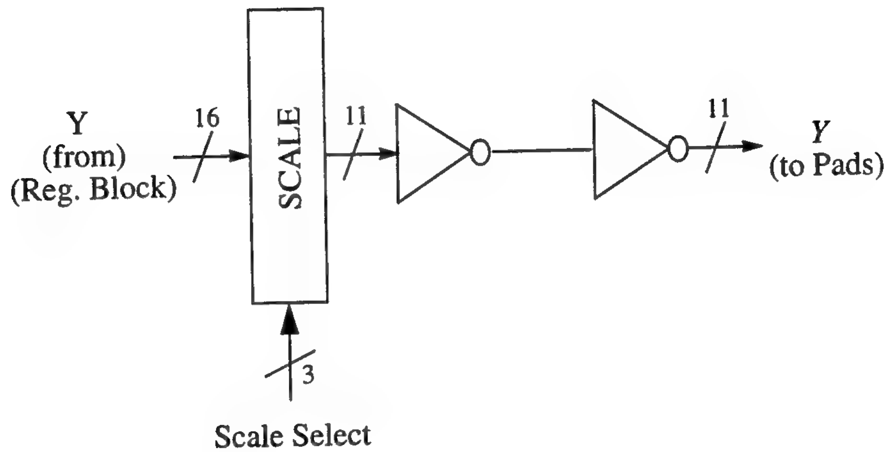


Figure 8 - Output data scaling.

Table 5: Scale Select vs. Y Output Data

Scale	11-bit *Y* output to pads										
000	Y[15]	Y[14]	Y[13]	Y[12]	Y[11]	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]
001	Y[14]	Y[13]	Y[12]	Y[11]	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]	Y[4]
010	Y[13]	Y[12]	Y[11]	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]	Y[4]	Y[3]
011	Y[12]	Y[11]	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]	Y[4]	Y[3]	Y[2]
100	Y[11]	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]	Y[4]	Y[3]	Y[2]	Y[1]
101	Y[10]	Y[9]	Y[8]	Y[7]	Y[6]	Y[5]	Y[4]	Y[3]	Y[2]	Y[1]	Y[0]

VII. Coefficient Memory

The coefficient memory is a static RAM block that stores 16 coefficients for input to the multiplier in the ALU. Each coefficient consists of a 13-bit 1X value, a 13-bit 3X value, and a 1-bit Shift that controls the multiplier output shift multiplexer in the ALU (see Section IX. for more information on the ALU architecture and the multiplier encoding scheme). The output shift provides for coefficients in the range of: $-2 \leq c < 2$ allowing for the implementation of the feedback multipliers in a second-order direct form II filter. The coefficient RAM is loaded through the 8-bit microprocessor bus interface. The coefficient loading is controlled by the input signals *Coeff*, *pmux[1:0]*, *addr[3:0]*, $\overline{\textit{data\_strobe}}$, and *din[7:0]*. The coefficient RAM output is controlled by a 4-bit read address supplied by the program memory (see Section VIII.). A block diagram of the coefficient RAM is shown in Figure 9. Figure 10 shows the write timing.

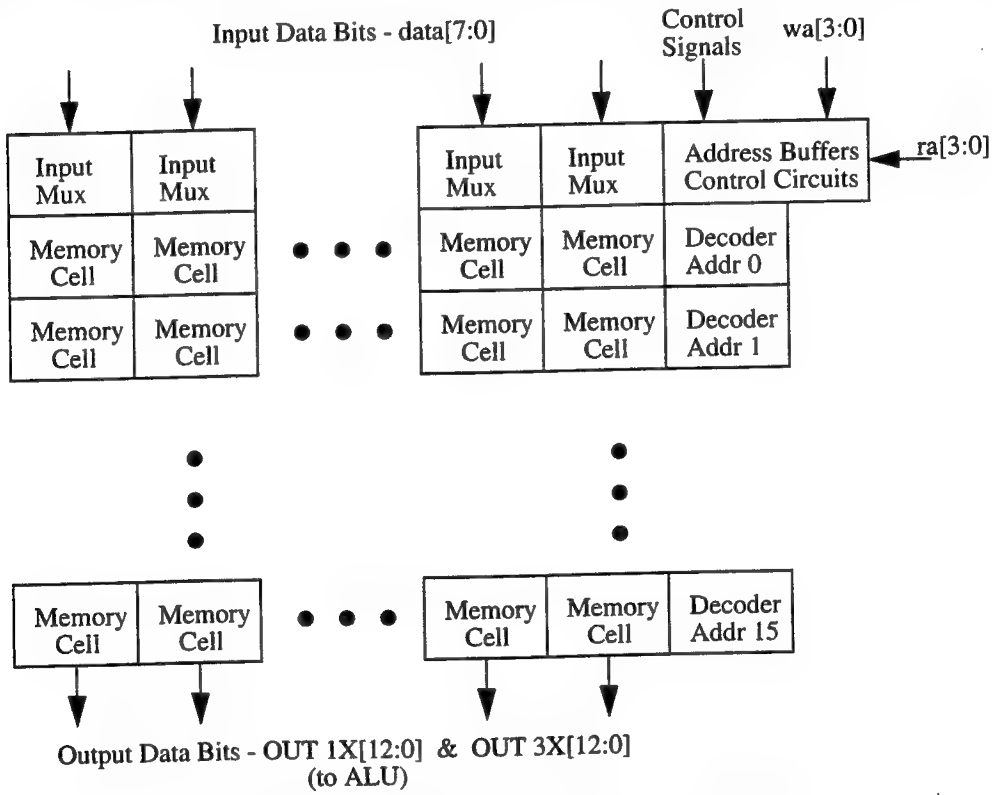


Figure 9 - Coefficient memory block diagram.

Table 6: Coefficient Memory Input Multiplexing

pmux[1:0]	*din[7]*	*din[6]*	*din[5]*	*din[4]*	*din[3]*	*din[2]*	*din[1]*	*din[0]*
00	IN1X[4]	IN1X[3]	IN1X[2]	IN1X[1]	IN1X[0]	------	------	Shift*
01	IN1X[12]	IN1X[11]	IN1X[10]	IN1X[9]	IN1X[8]	IN1X[7]	IN1X[6]	IN1X[5]
10	IN3X[4]	IN3X[3]	IN3X[2]	IN3X[1]	IN3X[0]	------	------	------
11	IN3X[12]	IN3X[11]	IN3X[10]	IN3X[9]	IN3X[8]	IN3X[7]	IN3X[6]	IN3X[5]

*This bit controls the Multiplier Output Shift Multiplexer in the ALU.

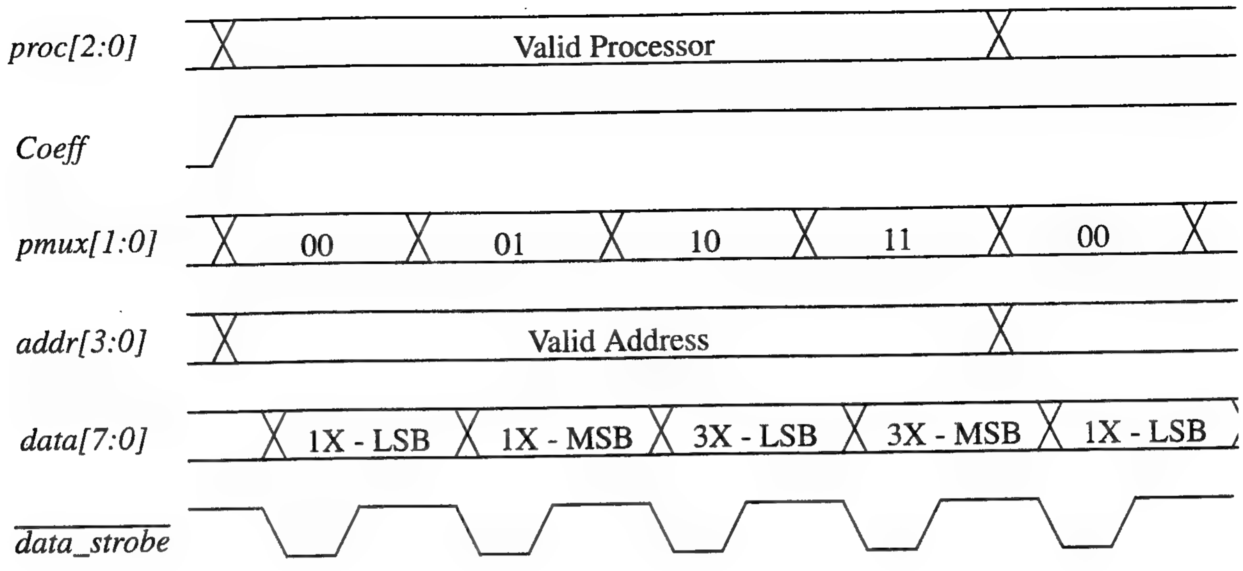


Figure 10 - Coefficient memory write timing.

VIII. Program Memory

The program memory is a static RAM block that stores up to 16 instructions. Each instruction is 32-bits wide, where the bits control the ALU data path (see Section IX.) and provide read and write addresses to the register blocks and coefficient RAM. Each processor has its own independent program memory. Program instructions are loaded through the 8-bit microprocessor bus interface. Two multiplexer control signals (*pmux[1:0]*) are used to select which 8-bit byte within the 32-bit instruction is being written. The instructions can be written randomly, but are read out sequentially. The address counter is incremented on the rising edge of phi1 and the instruction word is latched at the output of the RAM block on the rising edge of phi2. When reset (either internally or externally), the program address counter is forced to the stored reset address. The reset address can be changed at any time during operation through the microprocessor bus interface. Thus, multiple programs can be loaded and switched between during operation. For example, adaptive filters can be realized by programming two copies of the filter using different coefficient addresses. While the first copy of the filter is being run, the coefficients for the second copy can be updated from off-chip. Once updated, the coefficients can be "switched in" by changing the reset address to the start of the second program. Once switched, the coefficients associated with the first program can be updated. Figure 11 shows the write timing for loading program instructions, Figure 12 shows the reset timing, and Figure 13 shows the timing for changing the reset address during normal operation.

Table 7: Program Instruction Bit Functions

Bit	Name	Function	
31	Reset	Program Reset	
		0	No Reset
		1	Reset
30-27	regRW	Right Register Block Write Address	
26-23	regLW	Left Register Block Write Address	
22	selR	Right Output Bus Multiplexer Control (ALU)	
		0	Operand #1 (Op1)
		1	Adder Output (Sum)
21	selL	Left Output Bus Multiplexer Control (ALU)	
		0	Operand #1 (Op1)
		1	Adder Output (Sum)
20	selB	Add / Subtract (ALU)	
		0	Add
		1	Subtract

Table 7: Program Instruction Bit Functions

Bit	Name	Function	
19-18	selA	A Input to Adder - Multiplexer Control (ALU)	
		00	Multiplier
		01	Op1
		10	Zero
		11	NOT USED / INVALID
17-14	regRR	Right Register Block Read Address	
13-10	regLR	Left Register Block Read Address	
9-7	Op2	Operand #2 Multiplexer Control (ALU)	
		000	X Input Data
		001	Right Register Block's Output (RR)
		010	Left Register Block's Output (LR)
		011	Right Processor's ALU Output (RA)
		100	Left Processor's ALU Output (LA)
		101	Same Processor's ALU Output (A)
		110	Zero
		111	NOT USED / INVALID
6-4	Op1	Operand #1 Multiplexer Control (ALU)	
		000	X Input Data
		001	Right Register Block's Output (RR)
		010	Left Register Block's Output (LR)
		011	Right Processor's ALU Output (RA)
		100	Left Processor's ALU Output (LA)
		101	Same Processor's ALU Output (A)
		110	Zero
		111	NOT USED / INVALID
3-0	C-RA	Coefficient Read Address	

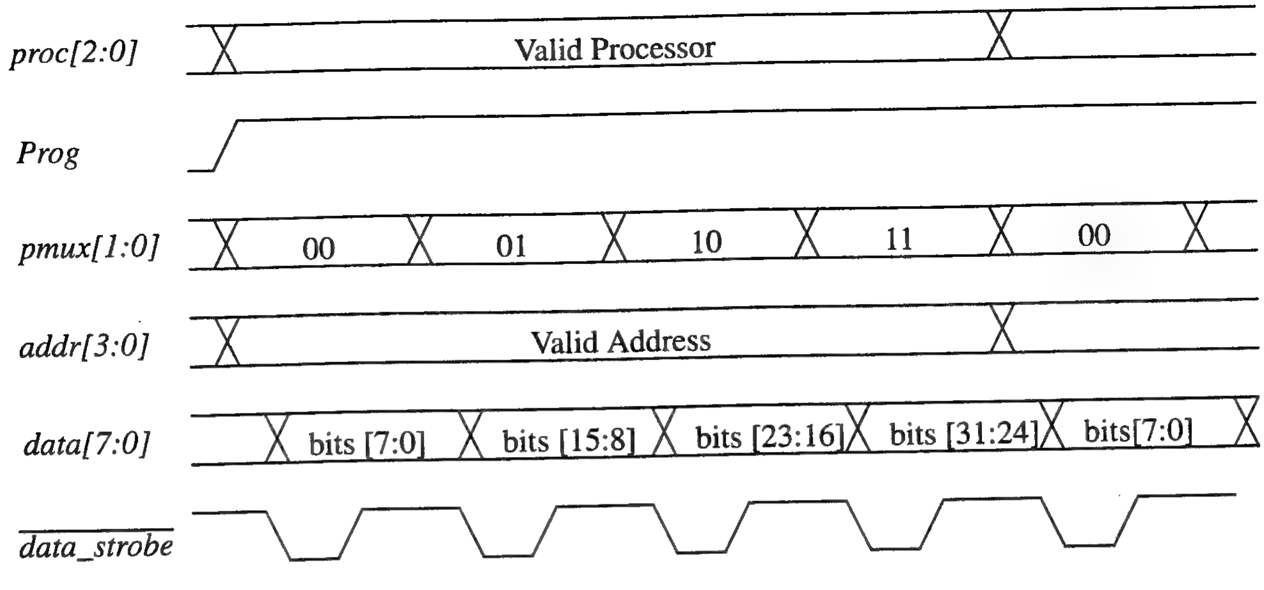


Figure 11 - Program memory write timing.

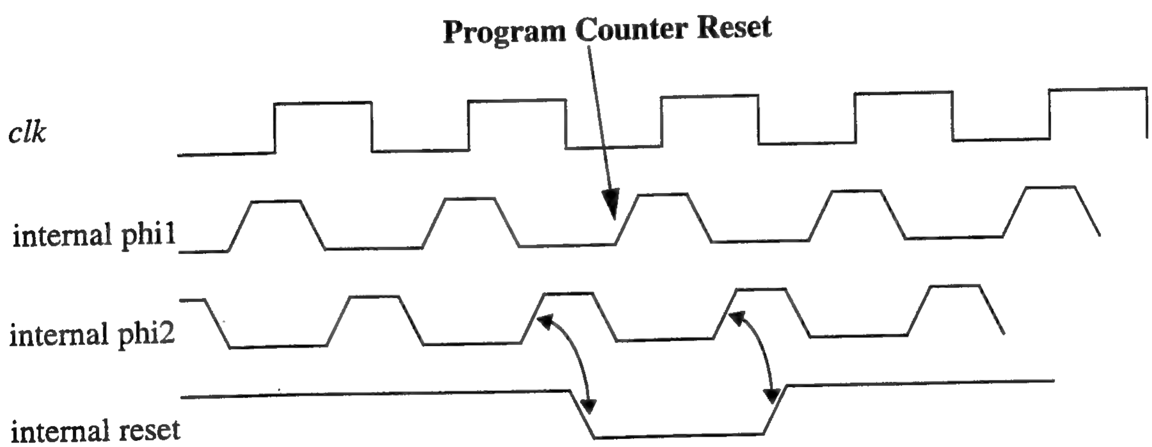


Figure 12 - Program memory reset timing.

Table 8: Program Memory Input Multiplexing

pmux[1:0]	*din[7]*	*din[6]*	*din[5]*	*din[4]*	*din[3]*	*din[2]*	*din[1]*	*din[0]*
00	prog[7]	prog[6]	prog[5]	prog[4]	prog[3]	prog[2]	prog[1]	prog[0]
01	prog[15]	prog[14]	prog[13]	prog[12]	prog[11]	prog[10]	prog[9]	prog[8]
10	prog[23]	prog[22]	prog[21]	prog[20]	prog[19]	prog[18]	prog[17]	prog[16]
11	prog[31]	prog[30]	prog[29]	prog[28]	prog[27]	prog[26]	prog[25]	prog[24]

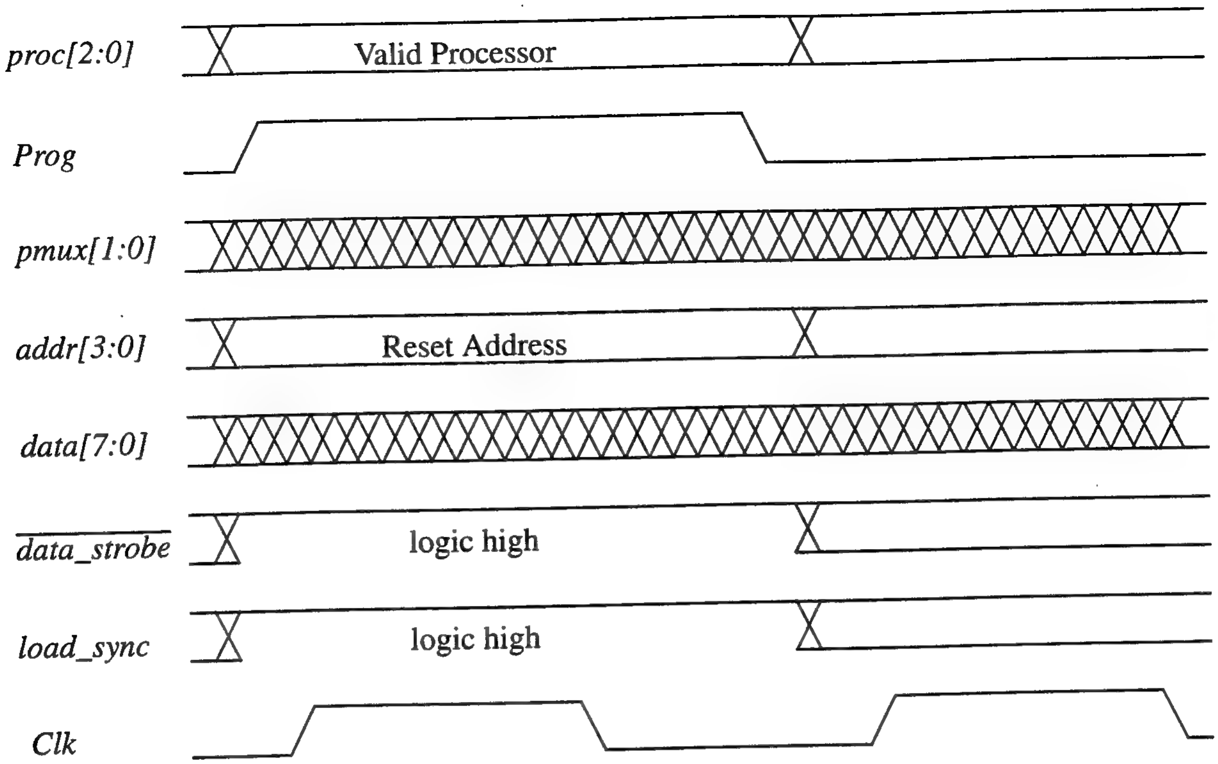


Figure 13 - Changing the program memory reset address during normal operation.

The *addr[3:0]* bus used to select the program or coefficient to be loaded can be synchronized to the internal clocks using the *load_sync* signal. This must be done when loading coefficients or changing the reset address while the chip is in normal operation. When initially loading coefficients or programs, the internal clocks are typically turned off using the $\overline{\textit{hold\_clk}}$ signal. During this mode of operation, the *addr[3:0]* bus synchronization register must be bypassed using the *load_sync* signal. Figure 14 shows a block diagram of the *addr[3:0]* bus synchronization circuitry.

Table 9: *addr[3:0]* Bus Synchronization Control

load_sync	MUX Output
0	*addr[3:0]* (directly from pads)
1	synchronization register output

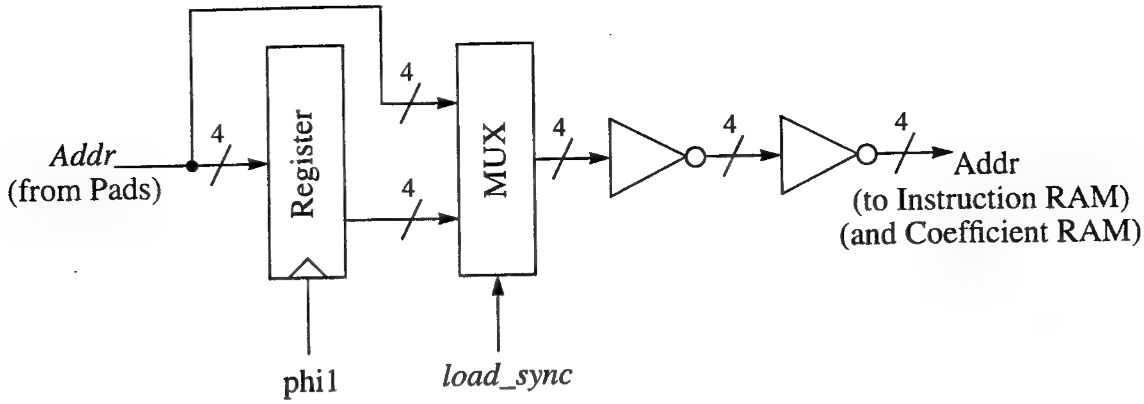


Figure 14 - *addr[3:0]* bus synchronization.

The outputs of the program memory are pipelined to match the delay through the ALU (see Section IX.) Thus, the register block read and write addresses for a given operation are stored within the same program word even though the actual read and write operations occur two clock cycles apart due to the pipeline delay through the ALU. Figure 15 shows the program memory output pipelining.

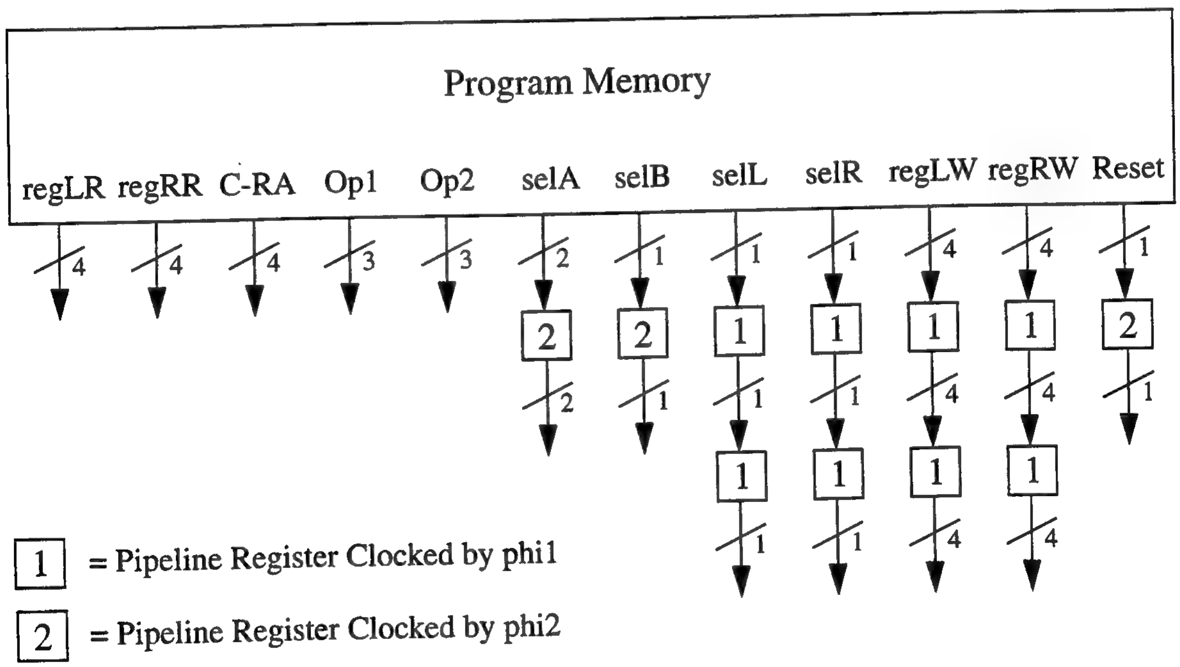


Figure 15 - Program memory output pipelining.

IX. ALU

The ALU is the "heart" of the processor, where all arithmetic computations are performed. Each processor's ALU contains an 11-bit by 11-bit hardware multiplier, a 16-word coefficient memory that provides one input to the multiplier (see Section VII.), a 16-bit adder / subtracter, and several multiplexers that control the data flow within the ALU. The ALU is pipelined so that the multiply operation occurs in one clock cycle. Thus, the ALU performs a multiplication and an addition simultaneously every clock cycle. All data inputs to the ALU are latched in at the rising edge of phi2. The ALU outputs are latched out at the rising edge of phi2. The multiplexer control signals that control the data flow within the ALU are provided to the ALU by the program memory at the rising edge of either phi1 or phi2 (see Section VIII.) A block diagram of the ALU is shown in Figure 16.

The operation of the ALU is as follows. First, two 7-to1 multiplexers select the two input operands (Op1 and Op2). Each operand is independently selected from the X input data (X), the right register block's output (RR), the left register block's output (LR), the right adjacent processor's ALU output from the previous clock cycle (RA), the left adjacent processor's ALU output from the previous clock cycle (LA), the current processor's ALU output from the previous clock cycle (A), or zero (Z). The multiplexer control signals (selOp1 and selOp2) are provided by the program memory a short time after the rising edge of phi2 (see Section VIII.) The multiplexer outputs are latched into the ALU at the rising edge of phi2, as shown in Figure 16. The Op1 input is then truncated to 11-bits and provided as one input to the multiplier. The second input to the multiplier is provided by the coefficient RAM (see Section VII.) The coefficient RAM is a 16-word static RAM that stores a 13-bit 1X value (the coefficient value) and a 13-bit 3X value (3 times the coefficient value). The coefficient memory provides an additional bit used to control a multiplexer at the output of the multiplier. This multiplexer allows the multiplier output to be shifted to the left by 1-bit (i.e., multiplied by 2) if desired. This capability is used to implement coefficients in the range of $-2 \leq c < 2$ for the feedback multipliers in a second-order direct form II IIR filter. In order to achieve high-speed operation in a small chip area, the multiplier was designed to take advantage of both 1X and 3X inputs. For a more detailed discussion of the multiplier refer to [2]. The coefficient memory read address is provided by the program memory shortly after the rising edge of phi2 (see Section VIII.) The coefficient memory outputs are latched into the multiplier on the rising edge of phi2, as shown in Figure 16. Since the multiplier's inputs and outputs are latched at the rising edge of phi2, the multiplier has the entire clock cycle to perform its computation.. The A input to the adder is supplied by the selA multiplexer (see Figure 16). It selects either Operand #1, the multiplier's output, or zero as input to the adder. The B input to the adder is either Operand #2 or the one's complement of Operand #2 (i.e., all bits inverted), selected by the selB multiplexer. For an addition operation, Operand #2 is selected. For a subtraction operation, the one's complement of Operand #2 is selected and the two's complement is formed by adding in the selB control signal to the adder's input carry. The adder input multiplexer control signals (selA and selB) are provided by the program memory at the rising edge of phi2 (see Section VIII.) The ALU provides outputs to both the left and right register blocks. These outputs can be individually selected as either the adder's output or Operand #1, as shown in Figure 16. The ALU's output is latched at the rising edge of phi2. The output select multiplexer control signals (selL and selR) are provided by the program memory at the rising edge of phi1 (see Section VIII.)

Additionally, the adder's output is made available as an input to both the left adjacent processor and the right adjacent processor. Thus, both processors have access to the result for use in the next clock cycle, effectively bypassing the dual-port register blocks.

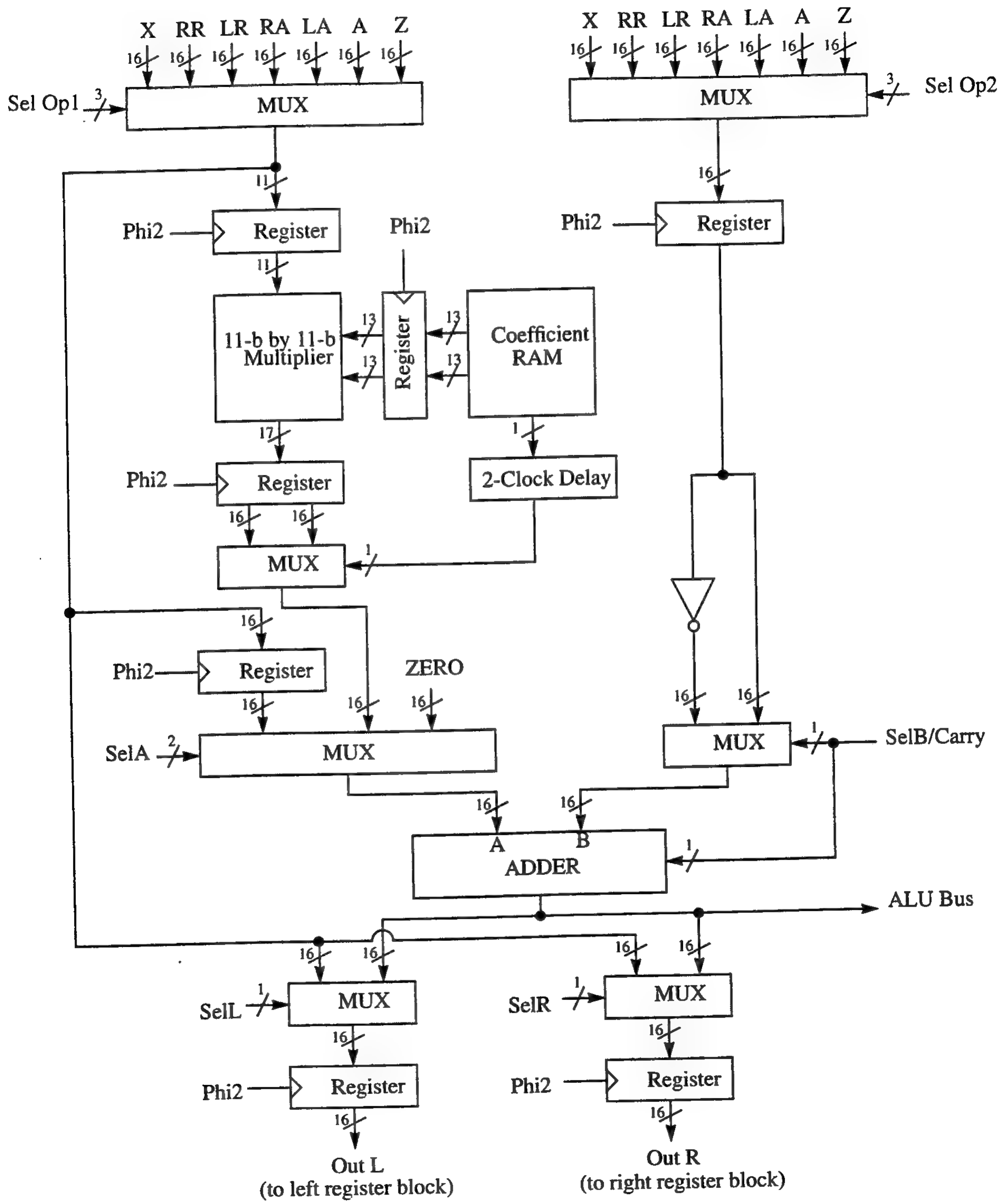


Figure 16 - ALU block diagram.

X. Dual-Port Register Block

The dual-port register block is a 16-word by 16-bit dual-port static RAM. Each dual-port RAM block is connected between two processors providing simple interprocessor communication. The register block has separate read and write data and address busses for each processor. Thus, each processor's access to the register block is completely independent. Automated programming techniques described in [1] are used to ensure that both processors do not write to the same memory location at the same time. The register block's read addresses are provided by the program memory at the rising edge of phi2. The register block's write addresses are provided by the program memory at the rising edge of phi1 (see Section VIII.) The register block's input data is provided by the ALU's output at the rising edge of phi2. The register block's output data is provided as input to the ALU. The ALU's input registers latch the data at the rising edge of phi2 (see Section IX.) The input data is written into the selected storage location during the time when phi2 is high. Since the write address is provided at the rising edge of phi1, the decoder outputs are stable before the write cycle begins. Separate input and output data buses are used to allow for high-speed operation. Figure 17 shows a block diagram of the dual-port register block.

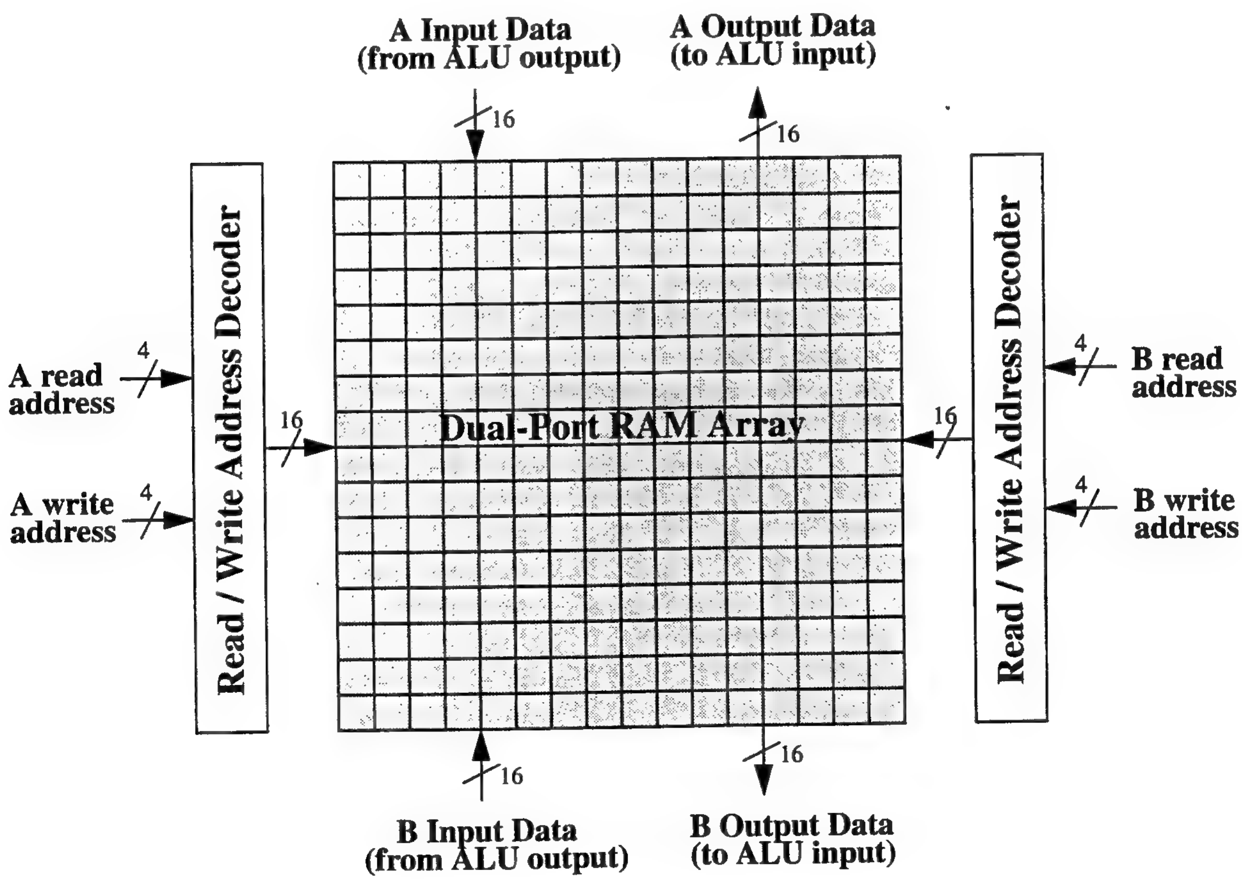
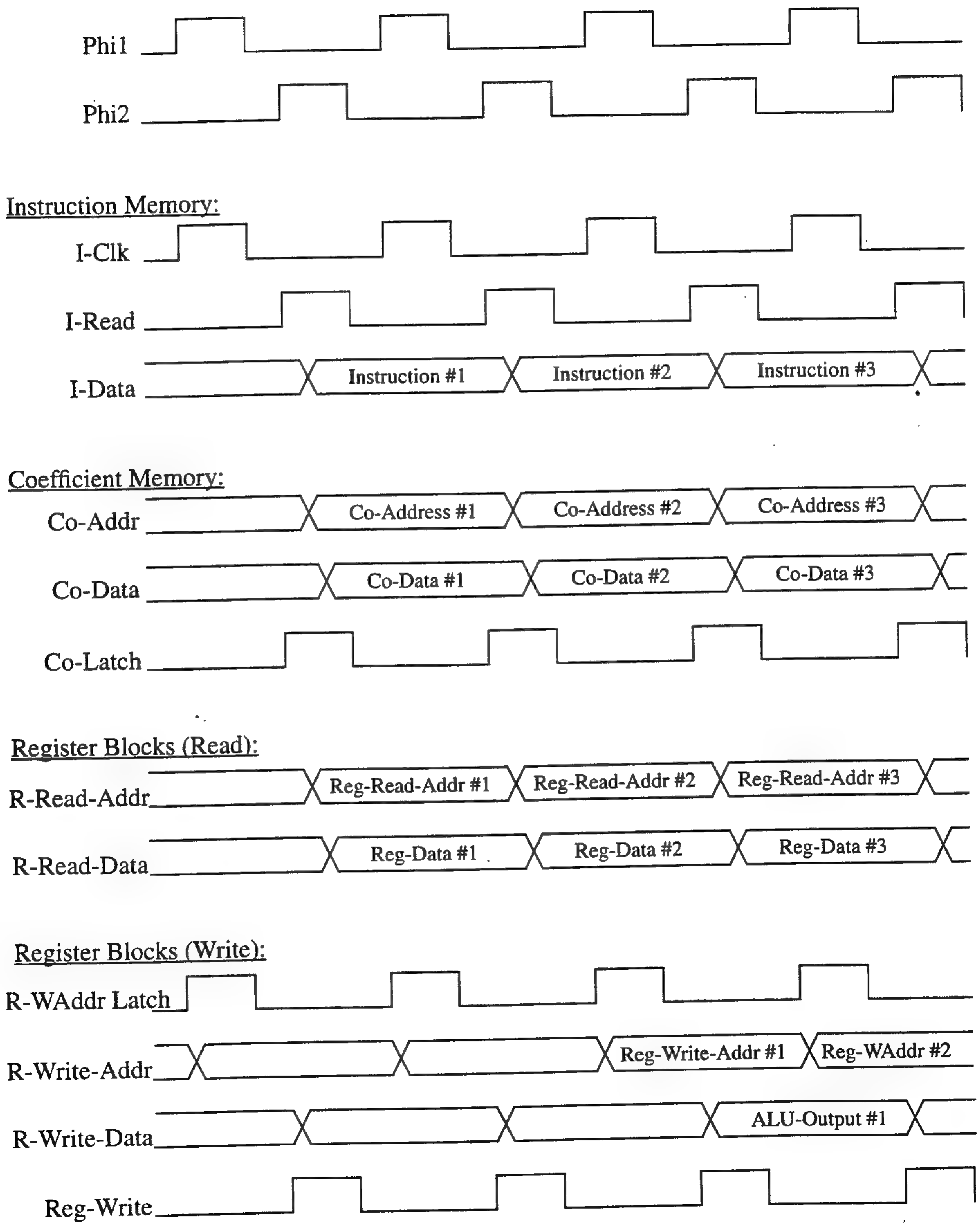
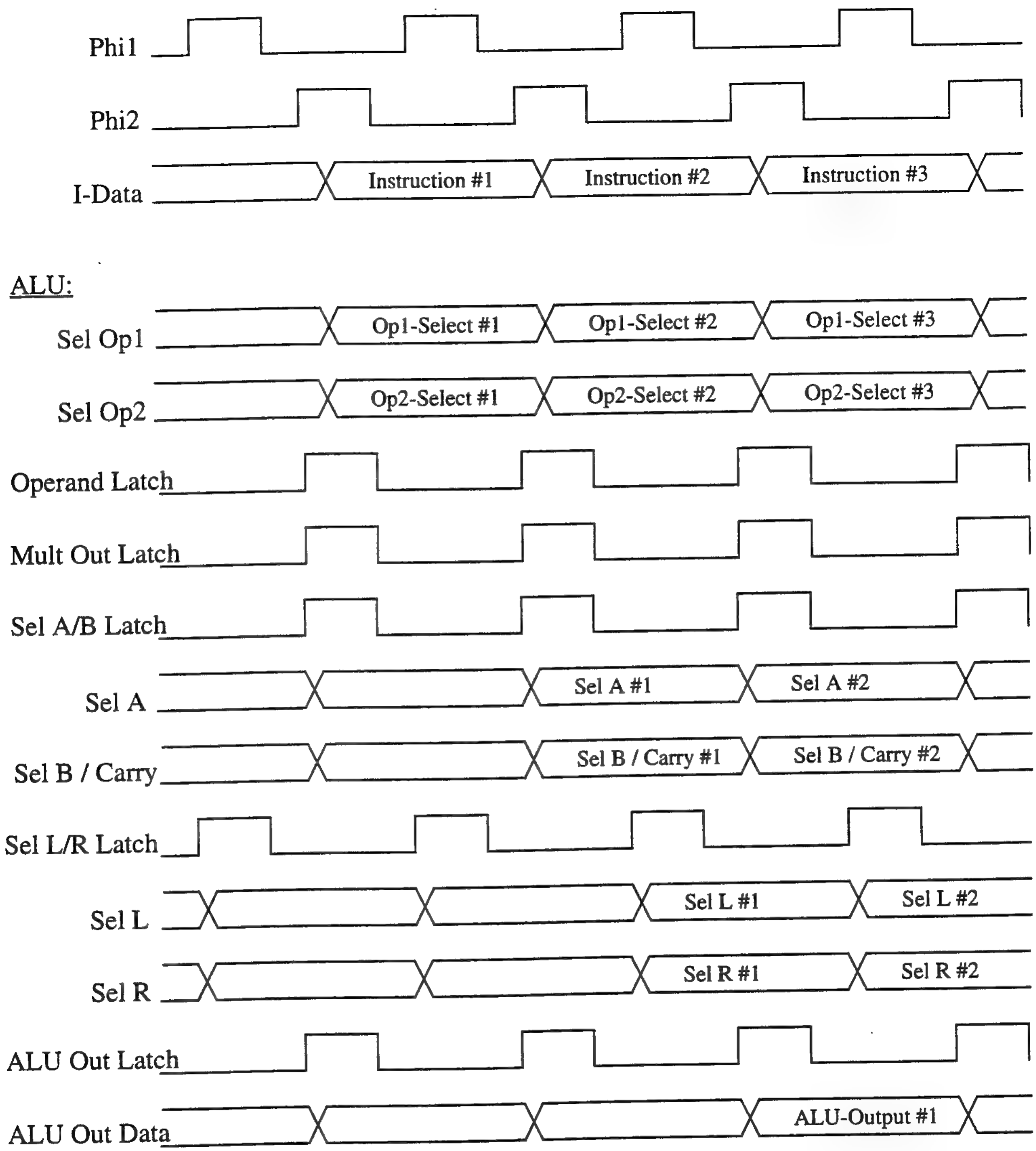


Figure 17 - Block diagram of the dual-port register block.

XI. Internal System Timing

The ring-processor system operates with a two-phase non-overlapping clock scheme (phi1 and phi2), as discussed in Section IV. Following is detailed timing information for all the major blocks within the system.




Phi1
Phi2
I-Data
Instruction #1
Instruction #2
Instruction #3
ALU:
Sel Op1
Op1-Select #1
Op1-Select #2
Op1-Select #3
Sel Op2
Op2-Select #1
Op2-Select #2
Op2-Select #3
Operand Latch
Mult Out Latch
Sel A/B Latch
Sel A
Sel A #1
Sel A #2
Sel B / Carry
Sel B / Carry #1
Sel B / Carry #2
Sel L/R Latch
Sel L
Sel L #1
Sel L #2
Sel R
Sel R #1
Sel R #2
ALU Out Latch
ALU Out Data
ALU-Output #1

XII. IC Testing Results

During the course of this project, several ICs were designed and fabricated to test out the major blocks in the ring-processor system before fabricating the complete five-processor system. Several of the test ICs were fabricated as MOSIS TinyChips. A TinyChip is a specially available option from MOSIS for 40-pin ICs of a prescribed size (2.25 mm by 2.22 mm including pads where the pads are in known locations) fabricated in 2-μm CMOS technology. This option is offered at a very low price because the pads are in known locations making packaging easier, only 4 packaged parts are returned, and no chip micrograph is taken. Due to the low price of this option, several of the test ICs fabricated for this project were designed as MOSIS TinyChips. All ICs were tested using a Tektronix LV500 IC tester. Although the LV500 tester is only capable of generating input test patterns at a maximum clock rate of 50 MHz (i.e., 20 ns cycle), it can control transition edges within a given clock cycle in 0.5 ns increments. Thus, it is possible to test circuits that operate at a clock rate higher than 50 MHz by including additional input and output registers around the test circuit that are clocked by separate clocks and then adjusting the timing between the two clocks within a given 20 ns LV500 test pattern cycle. This was the test methodology adopted for testing most of the ICs described below. Following is a brief discussion of each of the test ICs fabricated and the testing results.

A. Dual-Port Register Block Test IC

This IC was fabricated to test the dual-port register block. It contains a 16-word by 16-bit dual-port RAM block, input and output data registers, and read and write address registers. All registers, included for testing purposes, are clocked independently to facilitate accurate measurement of the read and write timing, as discussed previously. Due to the pad limitations of MOSIS TinyChips, only 3 input bits and 3 output bits were brought out to the pads for observation. Figure 18 shows a block diagram of this test IC. The register block core contains 4,064 transistors in a chip area of 1.66 mm^2 and was fabricated through MOSIS in a 2-μm CMOS P-well technology (TinyChip). All 4 parts received from MOSIS were fully functional with a worst-case read time of 15 ns and a worst-case write time of 16.5 ns.

Table 10: Dual-Port Register Block IC Testing Results

	Functional Test	T_{read}	T_{write}
Chip #1	passed	14 ns	15.5 ns
Chip #2	passed	15 ns	16.5 ns
Chip #3	passed	15 ns	16.5 ns
Chip #4	passed	15 ns	16.5 ns

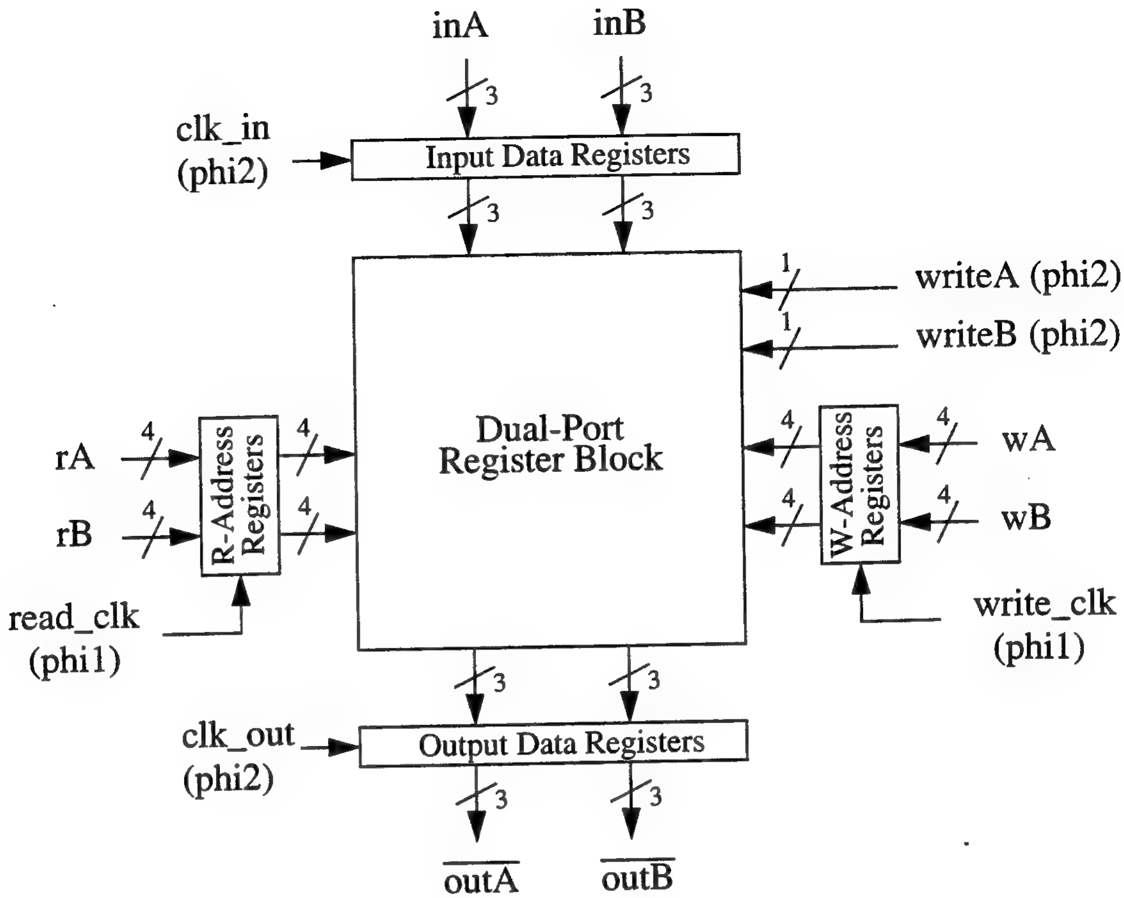


Figure 18 - Block diagram of the dual-port register block test IC.

B. 11-bit by 11-bit Multiplier Test IC

This IC was fabricated to test and characterize the multiplier used in the ALU. It contains the 11-bit by 11-bit multiplier, the coefficient and input data registers, the output data registers, and RAM to store the coefficient and input data. Figure 19 shows a block diagram of this test IC. The registers and RAM were included to try to accurately model the environment that the multiplier would see within the ALU (i.e., loading, drive capability, etc.) Separate input and output register clocks were provided to facilitate accurate testing of the multiplier delay (as discussed above). The multiplier core contains 3,492 transistors in a chip area of 1.53 mm^2 (1.313 mm by 1.116 mm) and was fabricated through MOSIS in a 2-μm CMOS N-well technology (TinyChip). SPICE simulations indicated a worst-case operating time of 22.5 ns (including the delay of the input registers). All 4 parts received from MOSIS were fully functional with a worst-case operating time of 23 ns. Testing results are given in Table 11 and Figure 20 shows the layout of the IC. For more detailed information about the multiplier refer to [2].

Table 11: 11-bit by 11-bit Multiplier IC Test Results

	Functional Test	Multiply Time (includes input register delay)
Chip #1	passed	23.0 ns
Chip #2	passed	22.5 ns
Chip #3	passed	23.0 ns
Chip #4	passed	23.0 ns

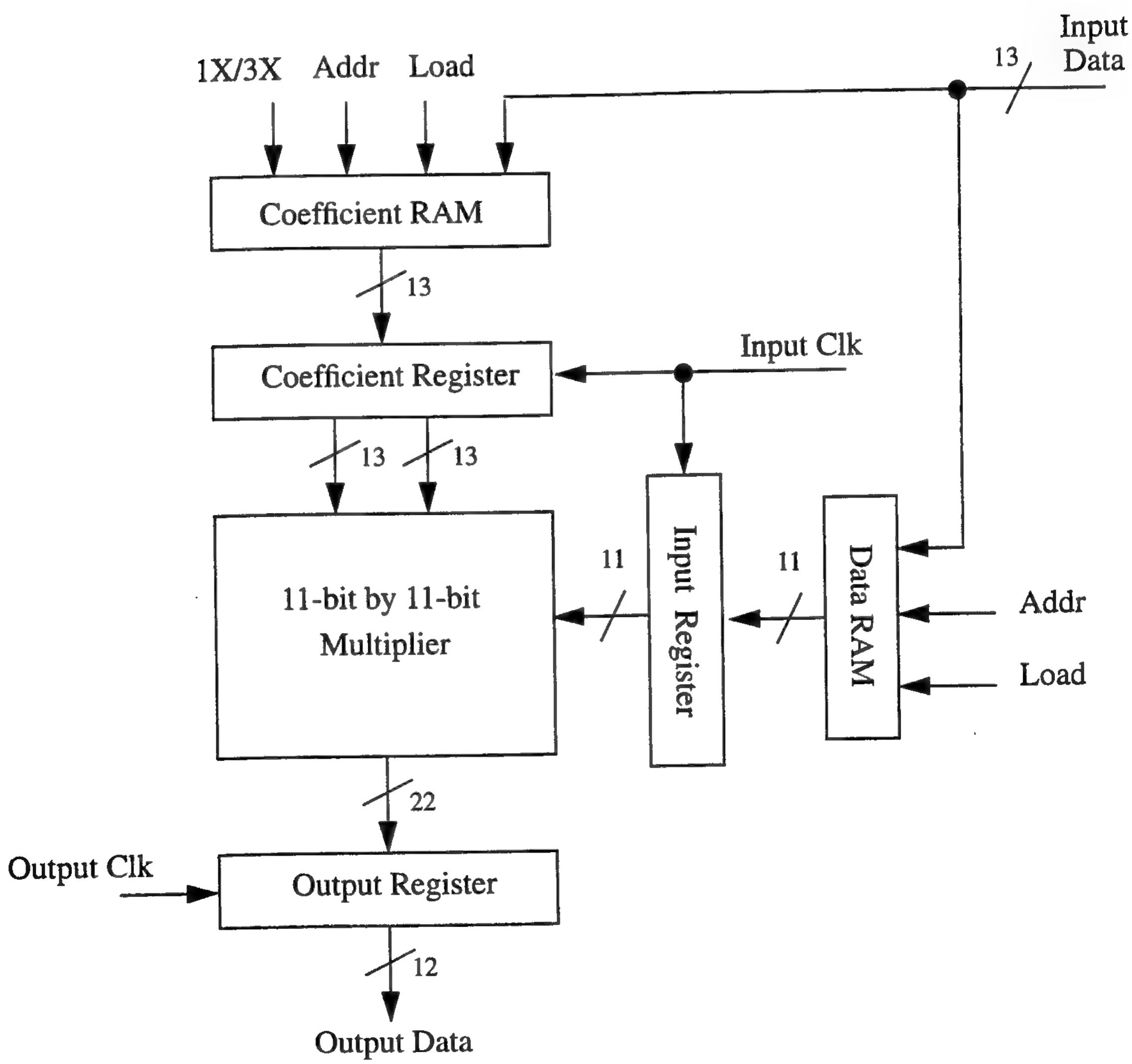


Figure 19 - Block diagram of the 11-bit by 11-bit multiplier test IC.

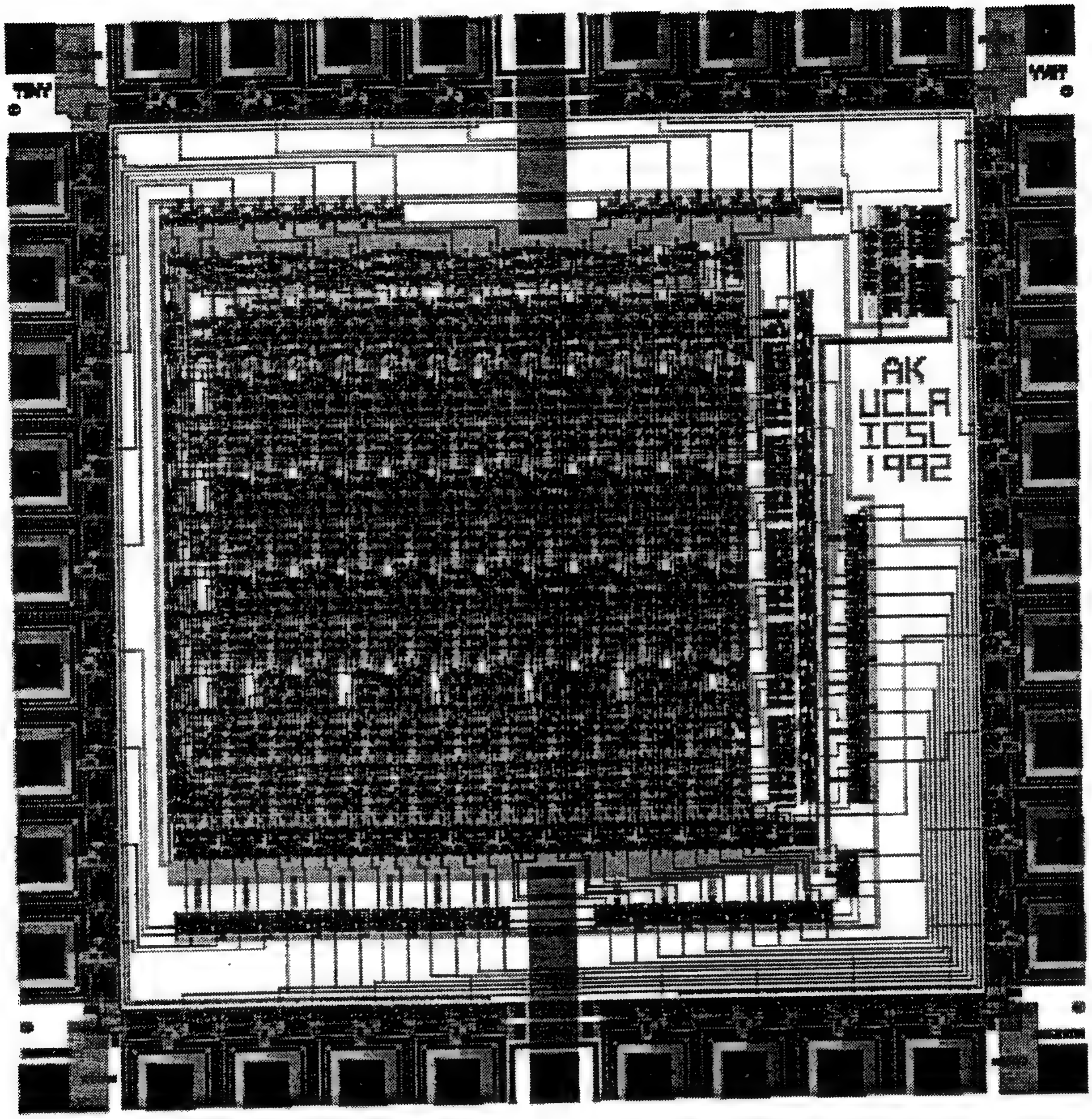


Figure 20 - Layout for the 11-bit by 11-bit multiplier test IC.

C. 11-bit by 16-bit Multiplier Test IC

This IC was fabricated as an extension to the 11-bit by 11-bit multiplier. It uses a 3rd order recoding scheme (as opposed to the 2nd order recoding scheme used in the previous multiplier) to extend the data precision to 16-bits while using the same number of partial products. This is achieved by replacing the 4-to-1 multiplexers used in the 11-bit by 11-bit multiplier with 8-to-1 multiplexers. The test IC includes input data and coefficient registers, RAM to store the coefficient and input data, and output data registers. The input and output data registers are clocked by different clocks to facilitate high-speed testing, as described previously. Figure 21 shows a block diagram of the test IC. For more information on the multiplier refer to [2]. The multiplier core contains 5,035 transistors in a chip area of 0.9 mm^2 (0.88 mm by 1.05 mm) and was fabricated through MOSIS in a 1.2-μm CMOS N-well technology. SPICE simulations indicated a worst-case operating time of 16 ns (including the register delays). Of the 24 parts received from MOSIS, 20 were found to be fully functional with worst-case operating times ranging from 17.5 ns to 19 ns with a mean of 18.175 ns. Figure 22 shows the test results for 5V and 3V supply voltages and Figure 23 shows the chip micrograph.

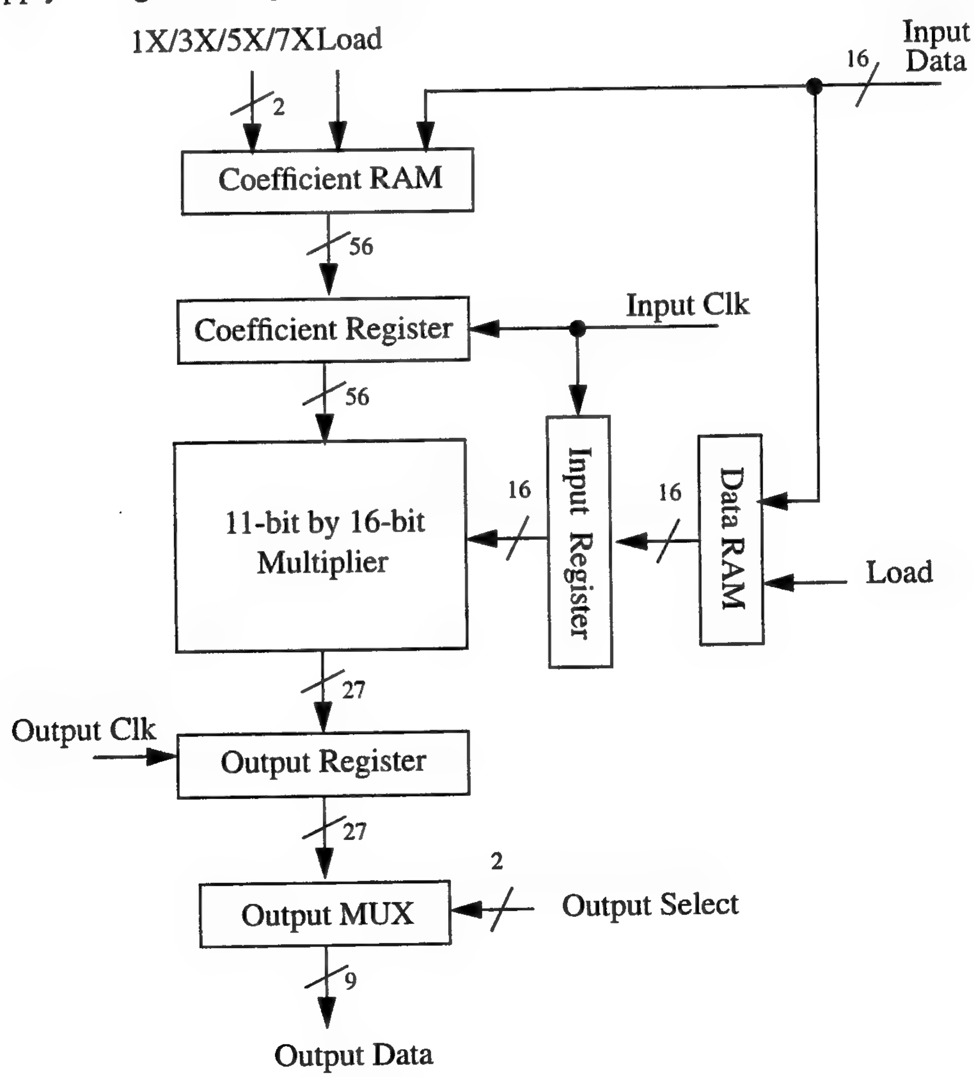


Figure 21 - Block diagram of the 11-bit by 16-bit multiplier test IC.

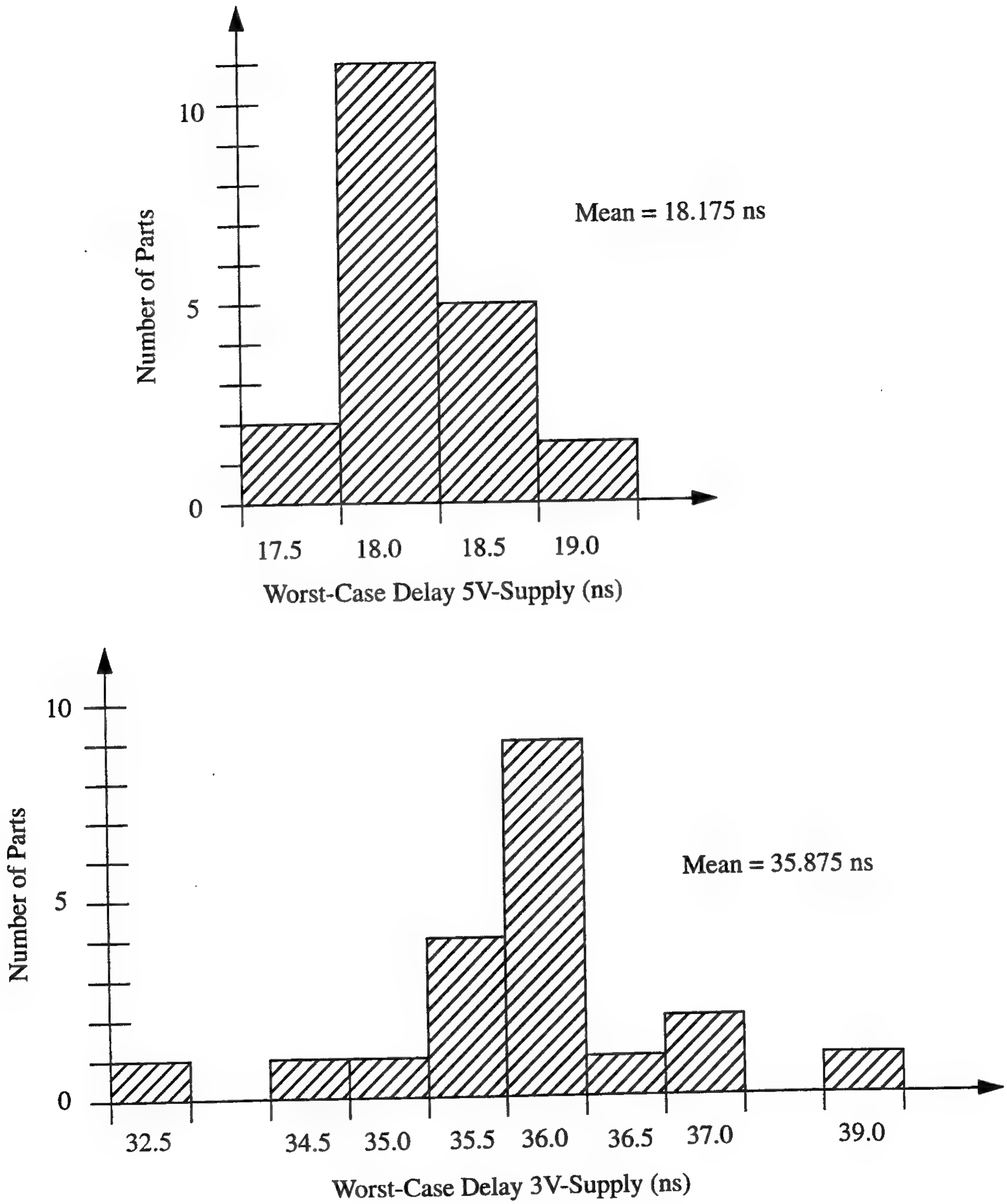


Figure 22 - Testing results for the 11-bit by 16-bit multiplier test IC.

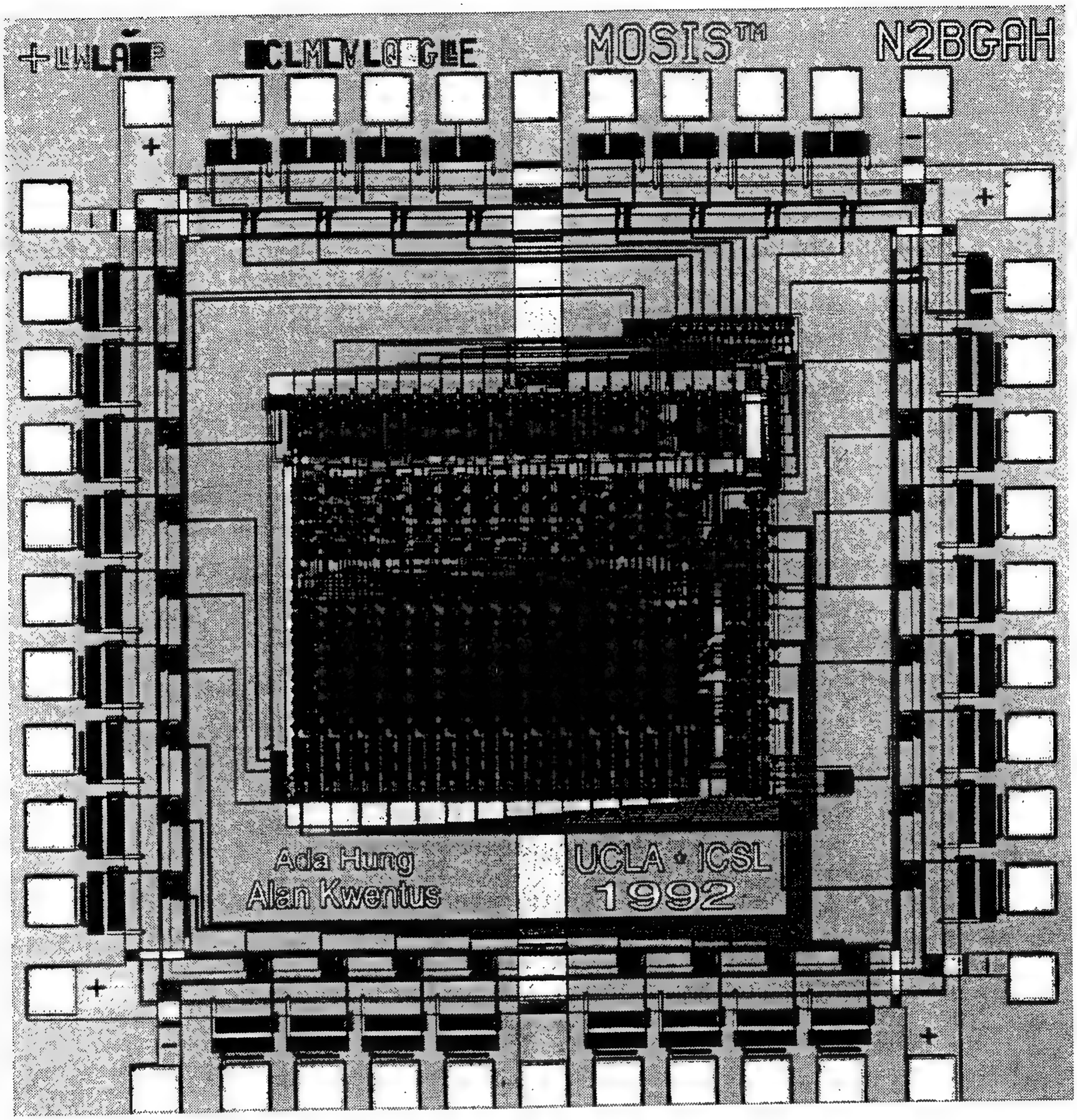
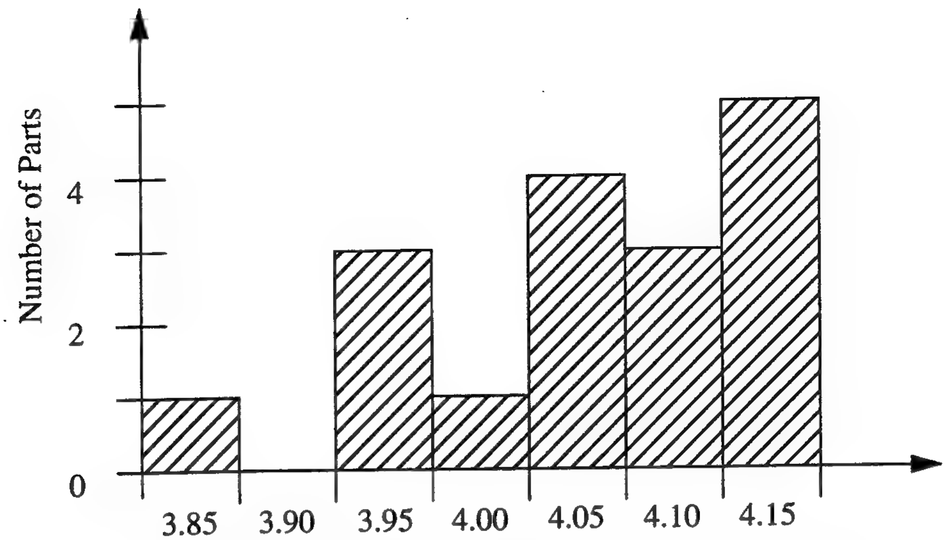


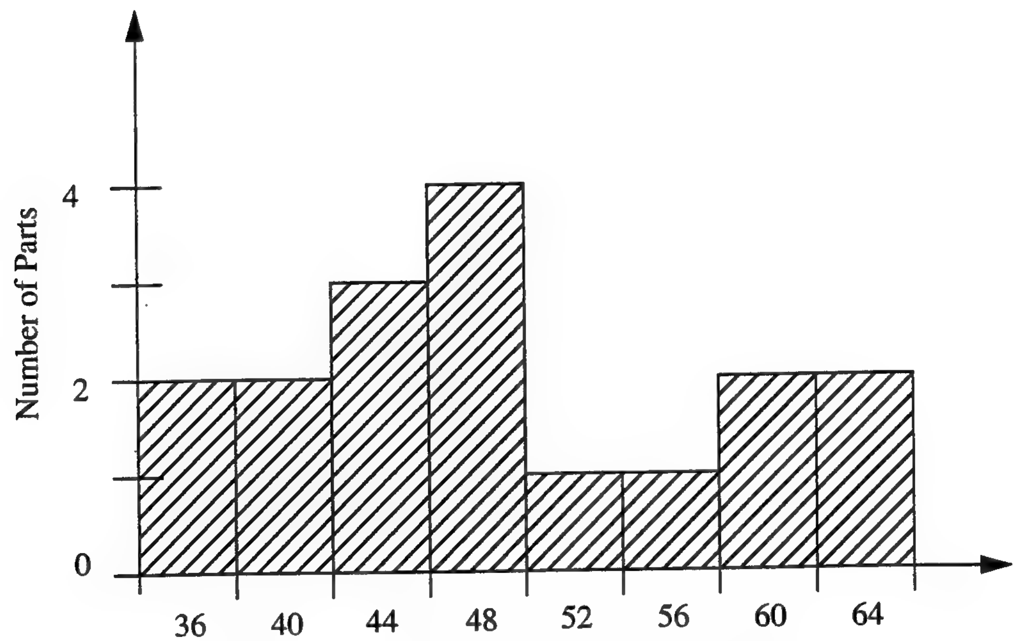
Figure 23 - Chip micrograph of the 11-bit by 16-bit multiplier test IC.

D. Single-Processor Test IC

This IC was fabricated to test out the major blocks of the ring-processor system before fabricating the complete five-processor ring. It consists of a single processor and two dual-port register blocks (basically, a one-processor "ring"). The chip provides for 11-bit input and output data with 16-bit internal data and 12-bit coefficients (stored in on-chip memory). The IC also has an 8-bit microprocessor bus interface for loading programs and coefficients. A block diagram of the IC is shown in Figure 2. The IC contains 24,723 transistors in a chip area of 14.8 mm^2 (3.7 mm by 4.0 mm including pads) and was fabricated through MOSIS in a 1.2-µm CMOS N-well technology. Of the 24 parts received from MOSIS, 19 were fully functional and all operated at a clock rate >50 MHz (the limit of the LV500 IC Tester). Figure 24 shows the minimum supply voltage for 50 MHz operation and the minimum clock period for a 3.3 V supply voltage. Due to limitations of the LV500 IC Tester, the minimum clock cycle period can only be tested in 4 ns steps. The single-processor IC was also programmed to implement several different filters. Figure 25 shows the transfer function of a 15-tap lowpass FIR filter which was run on the IC at a 50 MHz instruction clock rate. The filter requires 15 program steps so the data rate is 3.33 MHz (only 3 steps will be required on the five-processor ring so the data rate will be 16.67 MHz). Figure 26 shows testing results of the single-processor IC programmed to implement the 15th order lowpass filter for a two-tone input. The instruction clock rate is 50 MHz giving a sample rate F_s of 3.33 MHz and thus the input tones are at 333 KHz (normalized frequency 0.1) and 832.5 KHz (normalized frequency 0.25) respectively. The chip micrograph is shown in Figure 3.



(a) Minimum Supply Voltage for 50 MHz Operation (V)



(b) Minimum Clock Cycle for 3.3V-Supply (ns)

Figure 24 - Single-processor IC testing results.

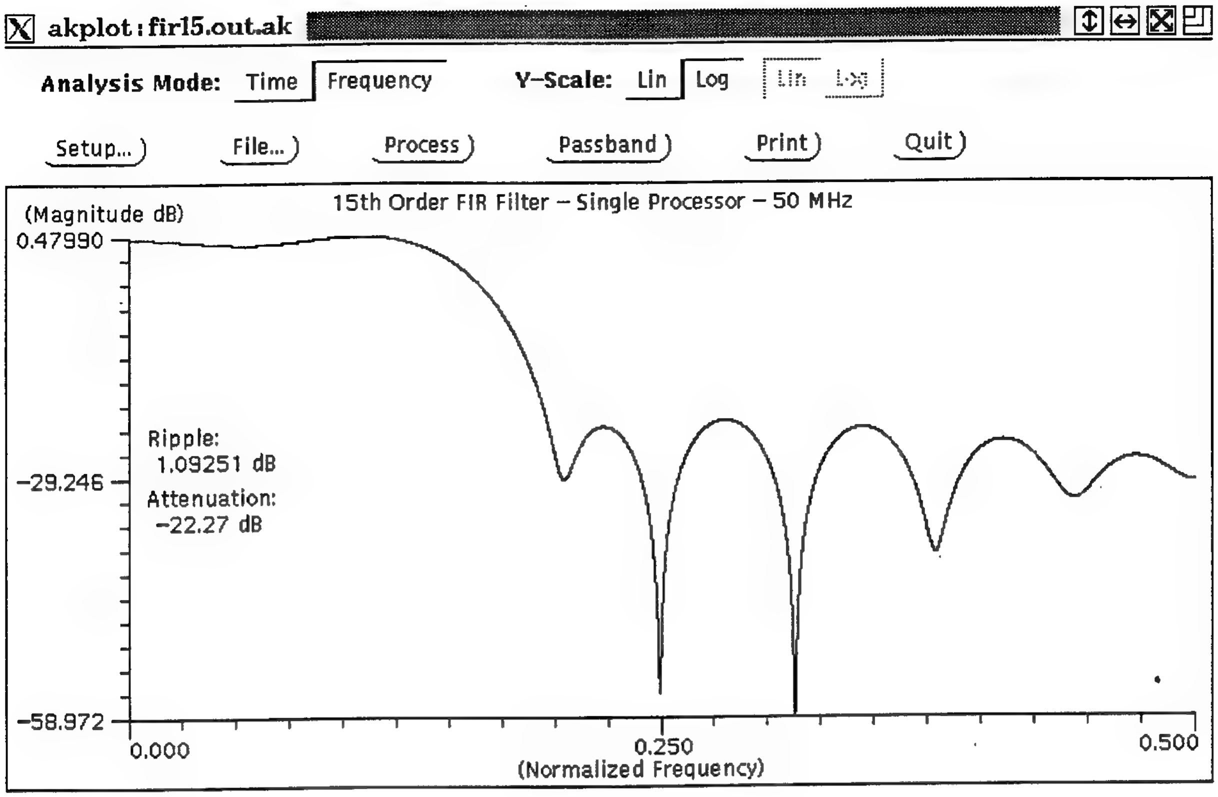


Figure 25 - 15-tap FIR lowpass filter transfer function.

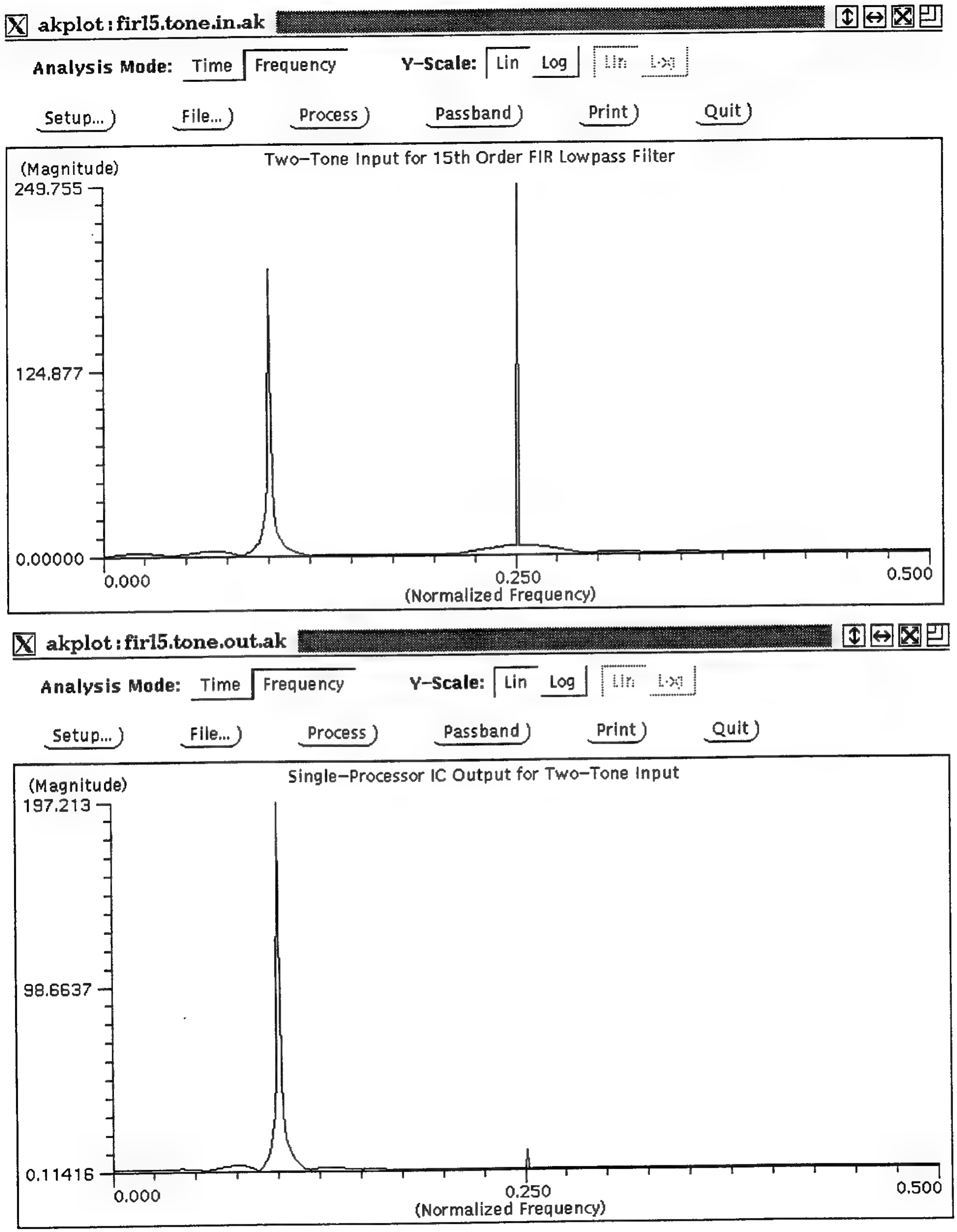


Figure 26 - Two-tone input and single-processor IC output for the 15-tap lowpass filter.

References

[1] A. Y. Kwentus, M. J. Werter, and A. N. Willson, Jr., "A programmable digital filter IC employing multiple processors on a single chip," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, pp. 231-244, June 1992.

[2] A. Y. Kwentus, H. T. Hung, and A. N. Willson, Jr., "An Architecture for High-Performance / Small Area Multipliers for Use in Digital Filtering Applications," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 2, pp. 117-121, February 1994.

An Efficient 180 MHz Programmable FIR Digital Filter

1 Introduction

FIR filtering is without a doubt one of the most important digital signal processing operations. In modern high-speed digital signal processing systems, data-rates of 100 MHz are becoming increasingly common. Implementing FIR filters at such high data-rates often requires the use of dedicated (non-programmable) custom application specific integrated circuits (ASICs). However, programmable FIR filters are required in many applications involving adaptive filtering, and they are often desirable for rapid prototyping, or for use in small volume applications where the cost of custom filter chips may be prohibitive. When implemented efficiently, a programmable filter can also be used instead of a custom FIR filter ASIC with advantages similar to those of FPGAs (i.e., it is an off-the-shelf standard product with no NRE costs and no inventory risk, it facilitates fast time to market, it is factory tested, and it allows design changes anytime). In addition to the increase in data-rates, another trend in high-performance signal processing systems is the increase in data word length. These factors, a longer word length and a higher data-rate, make the efficient implementation of a programmable FIR digital filter very challenging.

The implementation of high-speed programmable FIR digital filters (or correlators) is well researched. Invariably the transposed direct form FIR structure is used, with a separate multiplier for each[1] filter tap (i.e., each sample of the filter's impulse response). In such an implementation the data-rate is limited only by each filter tap's delay, which is largely the time required for a multiply and an add operation. The drawback, however, is the large chip area required to accommodate a large number of multipliers. Various methods to reduce the complexity and hence the area of the multipliers have been reported in the literature. In [1] serial multipliers are used, which severely limit the data-rate. In [2] an EPROM storing the products of all possible inputs by all filter coefficients is used in place of the multiplier. However, such intensive chip programming requirements severely limits its use as an adaptive filter. Advances in modern CMOS technology have also made possible a straightforward integration of a large number of standard multipliers on a single chip. For example, [3] reports a programmable filter chip consisting of 40 standard multipliers using 0.9-μm CMOS technology. However, this approach does not scale well with increasing word

[1]Or a separate multiplier for each pair of samples of the symmetric impulse response of linear-phase filters.

length since the area complexity of a standard multiplier varies as the square of the word length.

An effective method to reduce the complexity of the multipliers for the case of *dedicated* (non-programmable) FIR filters is to use the canonic signed-digit (CSD) [4–6] representation of the coefficient values. In essence, the CSD representation reduces the number of coefficient *digits* needed to represent each coefficient value, which correspondingly reduces the number of partial products produced when multiplying the input data by the coefficient values. This method, along with algorithms to design FIR filters with powers-of-two coefficients [6], results in the very efficient implementations of high speed dedicated FIR filters. Silicon compilers which produce the layout for such dedicated FIR filter chips using CSD coefficient representation are also readily available [7]. This approach, however, cannot be readily adapted to a programmable structure because neither the number of CSD coefficient digits nor the position of the individual CSD coefficient digits is known prior to programming.

In this paper we describe an effective solution to the problem of using the CSD approach for a programmable FIR filter (or correlator) structure, and we present [8] the first efficient implementation of a programmable linear-phase FIR digital filter using CSD coefficients. We show that it is possible to achieve high-speed processing while avoiding the severe hardware inefficiency that would result from a straightforward programmable tap implementation [1–3]. In a straightforward implementation many filter-tap "multipliers" would significantly waste valuable computational resources since all taps of a *programmable* structure would need to accommodate "difficult" coefficient values, while for any specific filter most taps would not require such extreme capabilities. For example, the taps whose coefficient values require higher precision are often located near the center of the impulse response of a typical lowpass FIR filter.

Our approach not only allows the programming of the number of filter taps and the specific filter-tap coefficient values, but it also provides the capability for programming the optimal allocation of hardware resources to each filter tap. Thus the computational resources that otherwise might have been wasted are made available to further increase the precision in any tap's coefficient representation, or for use in implementing a larger number of filter taps. We have achieved these unique advantages in our design by developing a novel switchable unit-delay. We have verified the ideas in a prototype chip that is capable of implementing a broad spectrum of linear-phase FIR filters employing up to 32 taps with 16-bit input and output data, in a die size of 5.9 mm by 3.4 mm using 1.2-μm CMOS technology. The prototype chip has been fabricated through the MOSIS service and tested to operate at data-rates as high as 180 MHz.

Section 2 briefly reviews the FIR filter and the signed-digit representation for numbers. It then introduces the programmable unit tap (p-tap) that is the basic element of our new programmable structure. Section 3 describes efficient circuits implementations for the programmable filter structure. Section 4 describes a prototype chip that implements a linear-phase FIR filter employing up to 32 taps with 16-bit input and output data and operating at data-rates as high as 180 MHz. Section 5 shows some design examples illustrating the advantages of our architecture.

2 Programmable FIR Filter Architecture

2.1 Review of FIR Filters and Correlators

The time-domain input-output relation for a causal Finite Impulse Response (FIR) system with impulse response $h(n)$ is given by the convolution formula

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k) \tag{1}$$

$$= h(n) * x(n) \tag{2}$$

where M is the length of the filter, $M-1$ is the order of the filter, and $*$ denotes the convolution operator. The minimum length M needed to implement a typical low-pass filter response is approximately proportional to the inverse of the normalized transition bandwidth of the filter's frequency response [9]. Therefore, for a programmable filter to be able to realize filter responses with sharp transition bands, we must allocate as large a number of taps M as possible to the programmable filter.

A mathematical operation that closely resembles convolution is correlation. For two signal sequences $x(n)$ and $y(n)$ each of which has finite energy, the crosscorrelation of $x(n)$ and $y(n)$ is a sequence $r_{xy}(n)$ given by

$$r_{xy}(n) = \sum_{k=-\infty}^{\infty} x(k)y(k-n) \tag{3}$$

$$= x(n) * y(-n) \tag{4}$$

It is obvious that an FIR filter (convolver) can be used as a correlator by simply reversing the ordering of the sequence $y(n)$ that the input data $x(n)$ is to be correlated with, and using that reversed sequence as the FIR filter coefficients. The system function of the FIR filter is obtained by taking the z transform of (1) which yields

$$H(z) = \sum_{n=0}^{M-1} h(n)z^{-n} \tag{5}$$

This can be written as a recursive equation:

$$H(z) = H_0(z) \tag{6}$$

with

$$H_k(z) = \begin{cases} h(k) + z^{-1}H_{k+1}(z) & \text{for } k = 0, \cdots, M-1 \\ 0 & \text{for } k > M-1 \end{cases} \tag{7}$$

Notice that each recurrence of (7) describes a single filter tap. That is, the output of the current tap $H_k(z)$ is the sum of two terms. One is the product of the input data and the filter coefficient $h(k)$, and the other is the output of the previous tap $H_{k+1}(z)$ after passing through a unit delay z^{-1}. Implementing $H(z)$ using (6) and (7) directly results in the well-known transposed (or inverted) direct form FIR structure shown in Fig. 1. (The index k in (7) advances from 0 to $M-1$ from right to left in Fig. 1.)

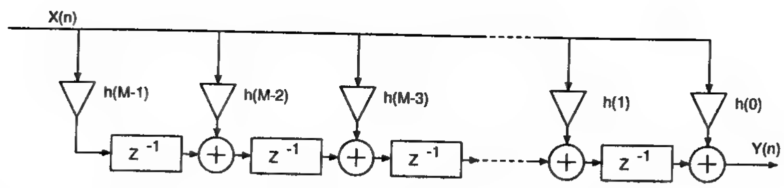


Figure 1: Transposed direct-form realization of FIR system.

2.2 Signed-Digit Representation

We use signed-digit representation to specify the filter coefficients. A radix-2 signed-digit fractional number C is represented by

$$C = \sum_{k=0}^{N} c_k 2^{-k} \tag{8}$$

where c_k is a signed-digit in the set {-1, 0, 1}, and C has a word length of $N+1$ digits. In general, the signed-digit representation for a given number is not unique. A minimal representation is one that requires the least number of nonzero digits. Among the minimal representations, there exists a unique representation known as the canonic signed-digit (CSD) representation for which no two nonzero digits are adjacent. The advantage of a minimal signed-digit representation such as CSD is that there are fewer nonzero terms in (8), which results in fewer partial products when the number C multiplies another number.

Algorithms for computing CSD coefficients for FIR filters that meet arbitrary specifications have been developed [6, 10, 11]. In general, these algorithms seek to limit the number of nonzero digits used to represent each signed-digit fractional coefficient value. That this is feasible in practice is demonstrated by the observation in [6] that only one nonzero digit in the CSD representation is typically required for each 20 dB of stopband attenuation in the filter specification, with an additional nonzero digit allocated to those impulse response coefficients whose magnitude exceeds 1/2. Thus a coefficient can be represented with a limited number of signed digits as

$$C = \sum_{k=0}^{L} c_k 2^{-p_k} \tag{9}$$

where c_k is a signed-digit in the set {-1, 0, 1}, and $p_k \in \{0, \cdots, N\}$. p_k now signifies the position of the signed-digit c_k. Notice that C can have up to $L+1$ nonzero digits and that its *effective* word length is still $N+1$ digits.

2.3 Programmable Unit-tap

The complexity of a programmable FIR filter is determined both by its length and by the number of nonzero digits allocated to each filter tap. As pointed out in the previous two sections, a filter with higher stopband attenuation demands a larger number of nonzero

digits for its coefficients, whereas a filter with a sharper transition band demands a larger number of filter taps. Clearly, satisfying both demands will tend to require a chip with an uneconomically large silicon area. Furthermore, either the large number of taps or the large number of coefficient digits would be wasted for filters with wide transition bands or low stopband attenuations, respectively. These wasted resources might otherwise be used to realize a filter with a larger number of taps or coefficient digits, whichever the application requires. The required precision for each coefficient is also non-uniform among all the coefficients. For example, the coefficient values that require higher precision are often near the center of the impulse response of a typical lowpass FIR filter. Furthermore, in the case of a correlator, it is uneconomical to allocate full precision to each tap because the average number of nonzero digits per tap may only be approximately $N/3$ [12]. In some correlator applications, many taps have zero value.

These difficulties can be overcome by having the number of nonzero digits allocated to each filter tap be one of the aspects of the chip's programming. This can be achieved by replacing the z^{-1} factor in (7) by a programmable factor z^{-q_k} so that the filter's transfer function becomes

$$H(z) = H_0(z) \tag{10}$$

with

$$H_k(z) = \begin{cases} C_k + z^{-q_k} H_{k+1}(z) & \text{for } k = 0, \cdots, M-1 \\ 0 & \text{for } k > M-1 \end{cases} \tag{11}$$

where $q_k \in \{0, 1, \cdots, Q\}$, and C_k is represented using (9) with up to $L+1$ nonzero digits,

$$C_k = \sum_{j=0}^{L} c_j 2^{-p_j} \tag{12}$$

We call the physical realization of each recurrence of (11) a *p-tap* to distinguish it from the filter tap in (7). We also call C_k the p-tap coefficient to distinguish it from the filter coefficient $h(k)$ in (7). L should be a small integer such that C_k is a low-precision number, allowing each p-tap to be implemented with minimal silicon area. Hence a large number of p-taps can be realized economically. When $q_k = 1$, the corresponding C_k of (11) is equivalent to the coefficient of an ordinary filter tap. Thus, a long filter that has a sharp transition band can be programmed with low-precision coefficients. When $q_k > 1$, $q_k - 1$ filter taps with zero coefficients are realized by a single p-tap. This is useful for implementing Qth band filters [13, pages 151-157], or for implementing a correlation sequence with many zero-value data. When $q_k = q_{k+1} = \cdots = q_{k+j-1} = 0$ and $q_{k+j} = 1$, the terms $C_k, C_{k+1}, \cdots, C_{k+j}$ are merged to form a single filter tap whose effective coefficient value $h(n)$ is

$$h(n) = \sum_{l=k}^{k+j} C_l \tag{13}$$

which has $j+1$ times the number of coefficient digits (i.e., precision) of a single p-tap. Thus, a filter with high-precision coefficients, for implementing a large stopband attenuation , can

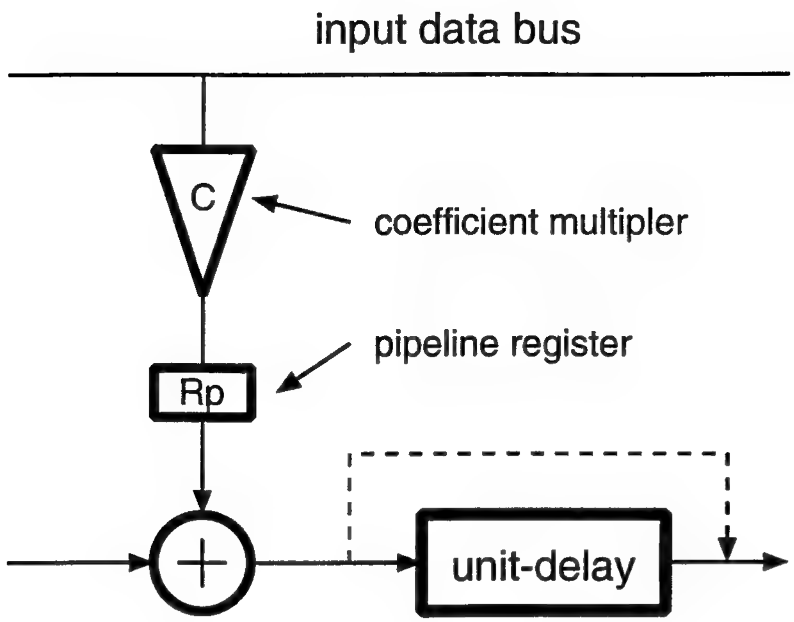


Figure 2: A p-tap.

be programmed by trading-off the total number of filter taps. Since the q_k are individually programmable, a large variety of filters can be programmed.

An example of a specific realization of a p-tap (the one implemented in our prototype chip) is shown in Fig. 2. In this example, the number of nonzero digits, $L+1$, in each p-tap is 2, and $q_k \in \{0,1\}$. The choice of having two coefficient digits per tap is partly due to the observation in [14] that the optimal (in efficiency) number of full adder stages between pipeline registers is two. The programmable z^{-q_k} term is implemented by a switchable unit-delay register which is turned on (not bypassed) when $q_k = 1$, and turned off (bypassed) when $q_k = 0$. This is indicated schematically by the dotted line in the figure. When the unit-delay is *on* the p-tap operates as a conventional filter tap. When *off* the summation node is connected immediately to the summation node of the next p-tap, merging the coefficient digits for the current p-tap and the next into a single filter-coefficient. If the unit-delay of the next p-tap is *on* then the current p-tap together with the next p-tap effectively forms a single filter tap that has twice the number of nonzero coefficient digits than that of a single p-tap. More nonzero coefficient digits can be added by combining additional p-taps in this manner. Fig. 3 illustrates three filter taps programmed to have 2, 4 and 6 coefficient digits.

2.4 Efficient Coding of Coefficients

In a programmable filter both c_k and p_k must be made programmable over the range $c_k \in \{-1,0,1\}$ and $p_k \in \{0,\cdots,N\}$. Notice that the fundamental property of the CSD representation, that no two nonzero digits are adjacent, would allow p_k to be programmed

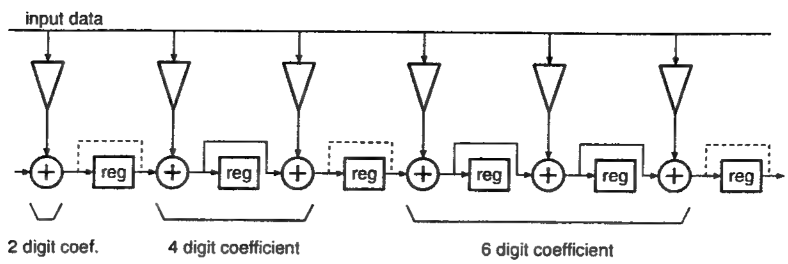


Figure 3: Filter taps programmed with different coefficient digits.

over a more restricted range:

$$p_k \in \{2k, 2k+1, 2k+2, \cdots, 2k+N-2L\} \tag{14}$$

for $k = 0, \cdots, L$. However, by using a full programmable range of $\{0, \cdots, N\}$, we can simplify the hardware required for storing and multiplying the coefficients, as will be shown shortly. Furthermore, if p_k is implemented by a programmable shifter using a series of multiplexors, very little if any silicon area would actually be saved by using the restricted range, due to the disruption of the regularity of the design. The only savings would be the smaller number of multiplexors needed.

In our implementation of the p-tap, as shown in Fig. 2, two coefficient digits are allocated to each p-tap, forming a coefficient:

$$C = c_0 2^{-p_0} + c_1 2^{-p_1}. \tag{15}$$

Since we permit both p_0 and p_1 to vary from 0 to N, the necessity to allow c_0 and/or c_1 to be zero can be eliminated by the following simple transformations: (i) If $C = 0$ is required, use:

$$0 = 2^{-p_k} + (-1)2^{-p_k}. \tag{16}$$

(ii) If the coefficient C requires only one nonzero digit c_k, we expand it into a two-nonzero-digit equivalent using one of the following representations:

$$c_k 2^{-p_k} = \begin{cases} c_k 2^{-(p_k+1)} + c_k 2^{-(p_k+1)} & \text{when } p_k < N \\ c_k 2^{-(p_k-1)} - c_k 2^{-p_k} & \text{when } p_k > 0. \end{cases} \tag{17}$$

Thus, the values required for each c_k now become {-1,1} instead of the conventional {-1,0,1} for the CSD representation. The elimination of the zero value simplifies the hardware for coefficient multiplication and reduces the storage requirements for the coefficient digits (a single bit, instead of two, is now sufficient to represent each coefficient digit c_k). A similar transformation can be made to eliminate the zero digit for implementations of a p-tap with more-than-two-digit coefficients.

3 Circuit Implementation

The effectiveness of our new programmable architecture depends upon the efficient implementation of the switchable unit-delay, the adder, and the coefficient multiplier. These will be discussed in the following sub-sections.

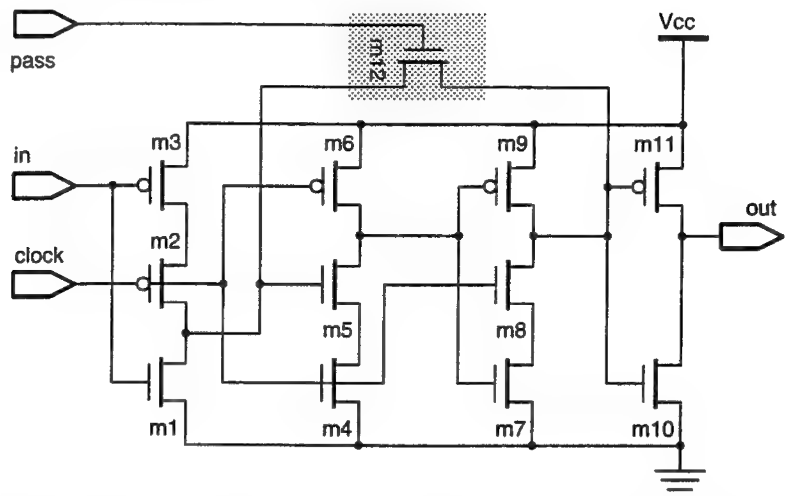


Figure 4: Schematic of the switchable unit-delay register.

3.1 Switchable Unit-Delay

We use a single-phase edge-triggered clocking scheme to simplify on-chip clock distribution and because it has been shown that high speed single-phase clocking can be achieved in CMOS circuitry [15]. Fig. 4 shows our circuit for the switchable unit-delay, which is identical to the true single-phase latch in [15] except for the N-MOS bypass-transistor m12. (This bypass can also be implemented with a full CMOS transmission gate with an additional P-MOS transistor.) With the addition of the single transistor m12, the unit-delay becomes switchable. When "pass" is low, the leading edge of the clock latches the data at "in." This is the normal unit-delay operation. When "pass" is high and "clock" is low, input data is passed through the input inverter (m1, m2, m3), through m12, and through the output inverter (m10, m11) to the output, thus disabling the unit-delay action. Notice that the clock signal must be disabled when "pass" is enabled. While this requires additional circuitry to disable the clock signal, this scheme has the overwhelming advantage of providing a simple switchable unit-delay circuit having no additional power dissipation due to an actively switching clock signal when the unit-delay register is bypassed. The additional circuitry to disable the clock signal is also insignificant because the clock signal is common to all the unit delays in a p-tap within the same data word. For example, in our prototype chip, a clock line is common to 80 registers.

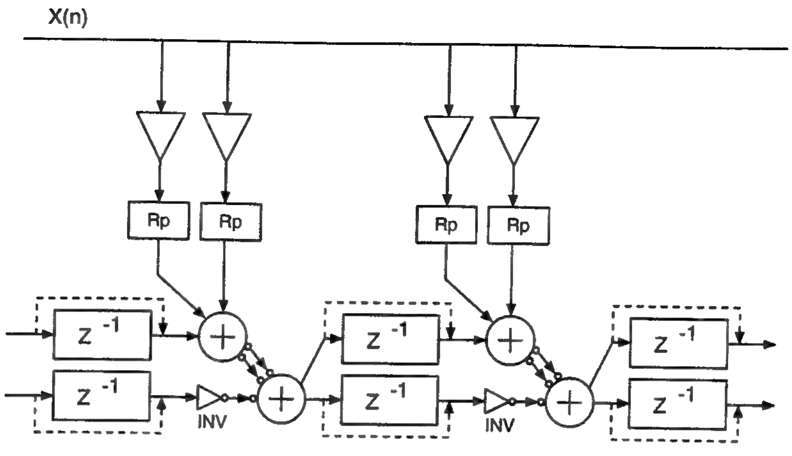


Figure 5: A section of p-taps implemented with carry-save adders.

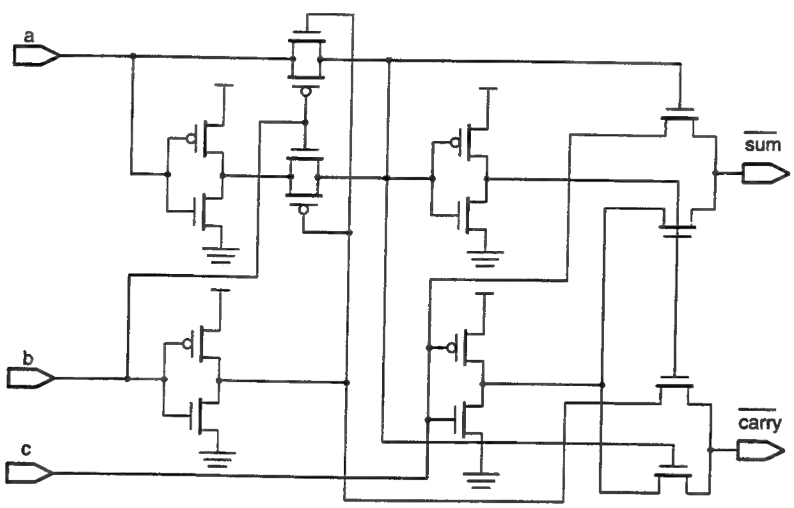


Figure 6: Schematic of the transmission gate adder.

3.2 Adder

Carry-save additions are used for the summation node in each p-tap to avoid the carry-ripple delay. With a two-digit p-tap coefficient, two partial products are produced by the multiplication of the input data and the p-tap coefficient. Because carry-save addition is used, the data sample from the previous p-tap consists of both the sum and carry outputs, therefore the summation node in each p-tap needs to add together four terms. This requires the cascade of two full adders as shown in Fig. 5

The adders are implemented with CMOS transmission gates as shown in Fig. 6. Both the carry and the sum outputs are inverted to eliminate output inverters, which reduces the transistor count as well as the adder delay. Since an adder with inverted outputs and non-inverted inputs is equivalent to an adder with inverted inputs and non-inverted outputs, the cascade of two inverted output adders restores the correct output polarity. However, the

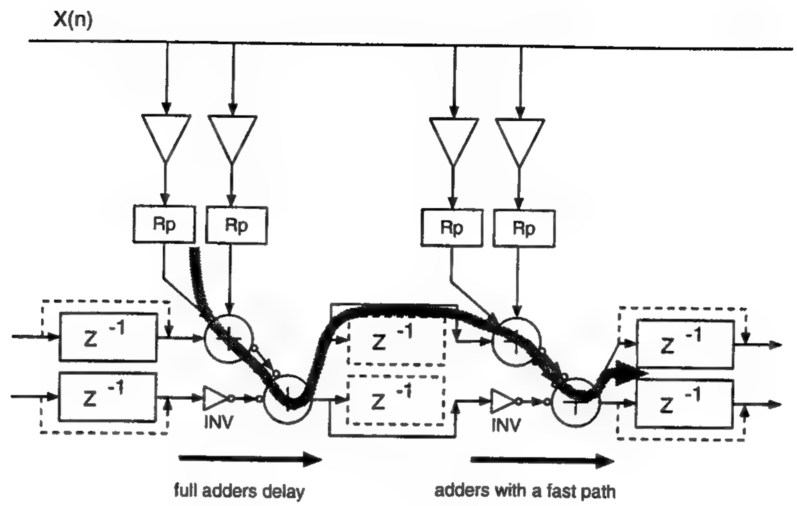


Figure 7: Critical path of a filter tap with two p-taps.

signal path that does not pass through both adders requires an additional inverter (INV) as shown in Fig. 5. Since the inverter is not in the critical path of the cascaded adders, it does not degrade the speed performance. Notice that the outputs of the transmission gate adders have reduced logic-high voltage levels due to the threshold voltage drop of the N-transistor pass gates. The voltage level is, however, restored by the programmable unit-delay register before feeding to the next adder stage.

The adder has a very fast signal path from its C input to both its carry and sum outputs. The delay from this "fast" input is only one transmission gate delay to the sum output, and a transmission gate delay plus an inverter delay to the carry output. The presence of this "fast" input is used to improve the speed of the cascaded adders as follows. When two or more p-taps are merged together to form a filter tap, the adders are connected in series. However, since the partial products are computed simultaneously, the delay of the adder chain can be reduced by designing the full adder such that it has a fast path from one of its inputs to both its sum and carry outputs. By feeding the two partial products to the two "slower" inputs, the critical path delay for each pair of cascaded full adders is only a normal full adder delay plus the fast adder path. Therefore, the critical path for a filter tap is

$$\begin{aligned} t_{total} &= t_{adder} + (K-1)t_{adder_{fast}} + \\ &\quad (K-1)t_{unit-delay(off)} + t_{unit-delay(on)} \end{aligned} \tag{18}$$

where t_{adder} is the full delay through a pair of cascaded full adders, $t_{adder_{fast}}$ is the delay through a pair of cascaded full adders with a fast path, and K is the number of p-taps that are merged into a filter tap. This is illustrated in Fig. 7.

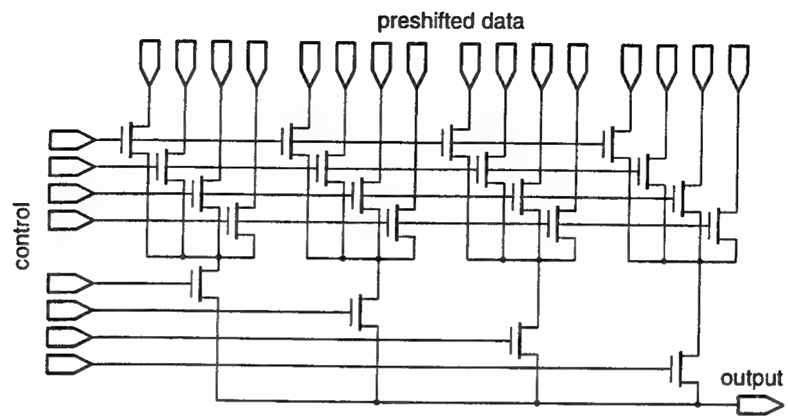


Figure 8: Schematic of the two-level NMOS multiplexor.

3.3 Coefficient Multiplier

Each coefficient digit $c_k 2^{-p_k}$ consists of two factors, one is the 2^{-p_k} weighting factor, which can be implemented by a right shift, and the other is the c_k bit multiplication factor.

The 2^{-p_k} shifting is realized by selecting one of 16 (the word length of the input data) hardwired preshifted data via two levels of 4-to-1 NMOS transmission gate multiplexors (Fig. 8). The advantage of the two-level multiplexing is the reduction in the number of control lines to eight. To save silicon area, each block of hardwired preshift is shared by four sets of multiplexors (or two p-taps, since each p-tap has two coefficient digits).

Since c_k is either 1 or -1, and never 0, multiplication for each digit is easily handled by an invert/no-invert circuit realized by a simple exclusive-OR gate. This forms the 1's complement of the shifted data for the case of a negative coefficient digit. The LSB of 1 that needs to be added to form the 2's complement negation is accumulated into a sum for all the coefficient digit multipliers. This sum forms part of the compensation vector that is added to the first p-tap in the forward datapath (which has free adder inputs) [7].

Due to the considerable delay incurred by the long input data bus and the two-level transmission gate multiplexor, a pipeline register (shown as Rp in Fig. 2) is inserted after the coefficient multiplier in order to obtain a higher maximum data-rate.

4 Prototype Chip

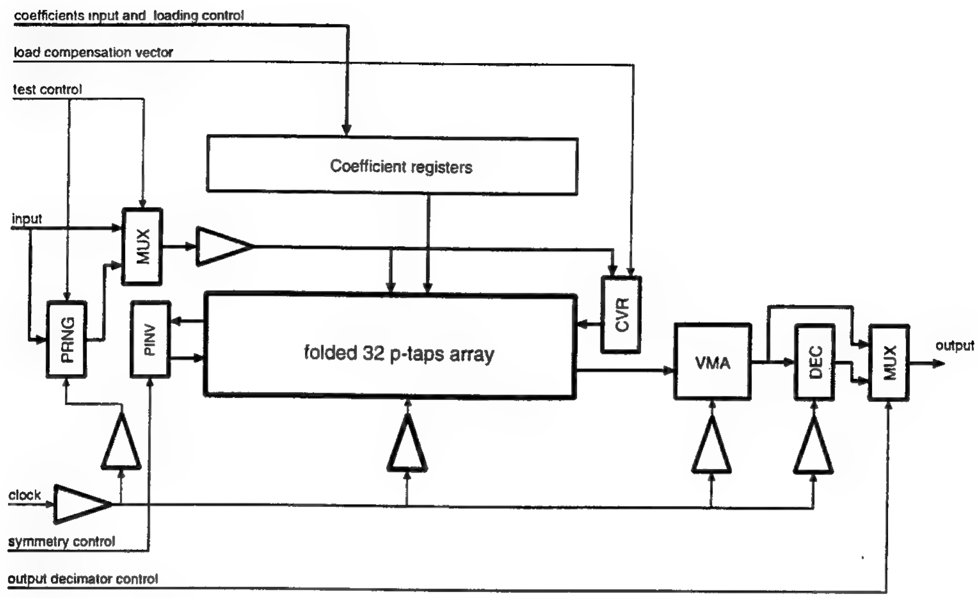


Figure 9: Block diagram of the programmable FIR chip.

Fig. 9 shows the block diagram of our programmable linear-phase FIR filter chip. It has 16-bit input and output data. Its internal word length is chosen to be 20-bit to ensure that the error at the filter's output due to internal quantization is less that the quantization error due to the finite word length of the data. The core of the chip is the series of 32 p-taps, folded to share the symmetrical coefficients for linear-phase operation. Surrounding the core are the clock and data drivers, the vector merge adder (VMA), the compensation vector register (CVR), the programmable inverters (PINV), the coefficient registers, and testing circuitry.

The carry and sum outputs from the last p-tap are added using a 20-bit VMA to produce the final output. The VMA is implemented by a five stage pipelined carry-ripple adder. The pipelining removes the VMA from the filter's critical path.

The programmable compensation vector register (CVR) is used to correct the filter core output by adding in the MSB sign-extension and the additional 1's needed for 2's complement negation. It can also be used to select between rounding or truncation. The compensation vector is programmed through the input data bus because of the limited number of pins (84) available on our small die. A programmable inverter (PINV) is inserted in the middle of the series of p-taps to permit the chip to implement filters with either symmetrical or anti-symmetrical impulse responses.

To facilitate the testing of the chip, a 16-bit pseudo random number generator (PRNG) and an output decimator (DEC) are implemented on-chip. The PRNG is based on the type 2 linear feedback shift registers [16, pages 432-441] which will produce a pseudo random number sequence provided that the states of the linear feedback shift registers are not identically

Table 1: Summary of the prototype programmable FIR chip.

Maximum FIR order	32
Technology	1.2-μm CMOS
I/O word length	16-bit
Coefficient word length	16-bit
Internal word length	20-bit
Core area	4.2 x 2.8 mm^2
Die size (with pads)	5.9 x 3.4 mm^2
Maximum data-rate	180 MHz
Power Supply	5 V
Power consumption	1.3 W @ 180 MHz
Packaging	84-pin PGA

zero (which will produce a sequence of constant zeros). To avoid the zero state, the starting state of the PRNG is made to be programmable through the input data bus. The output decimator, when not bypassed, decimates the output samples by a factor of 16. In our test setup, when testing is performed within the frequency range of our tester (< 50 MHz), the output decimator is bypassed and input test vectors are applied by the tester. To perform testing beyond the frequency range of the tester, the clock signal to the chip is supplied by an external high frequency source, the PRNG is turned on, and the output is decimated and sampled asynchronously by the tester. A computer program is used to correlate the outputs sampled by the tester with the calculated result to verify the chip's functionality at the higher speed. This permits us to verify the core of the chip to at least 8 times the sampling speed of our tester.

The chip was designed using the Mentor Graphics GDT VLSI CAD tools. The leaf cells for the chip are all custom layouts, so as to obtain the best performance. The leaf cells are assembled by a compiler with parameterized word length and number of p-taps. Thus, any size filter chip can be generated very easily. The compiler is written in the Genie language, a C-like interpreted language with interface to access the GDT layout database. A summary of the prototype chip is given in Table 1. The prototype chip (Fig. 10) was fabricated through the MOSIS service using the Hewlett-Packard 1.2-μm CMOS N-well process.

The prototype chip has been tested to operate up to a data-rate of 180 MHz, for filter taps consisting of single p-taps (i.e., at most two nonzero CSD digits per filter tap). For the case of two p-taps merged to form a single filter tap (i.e., at most four nonzero CSD digits per filter tap), the chip will operate up to a data-rate of 90 MHz. However, the input data can also be applied to two programmable FIR filters, each having half the number of filter coefficient digits per filter tap. The outputs of these filters can then be added together by an additional adder. When configured this way, the maximum 180 MHz data-rate can be achieved for filters whose taps would require up to four non-zero CSD digits. This concept

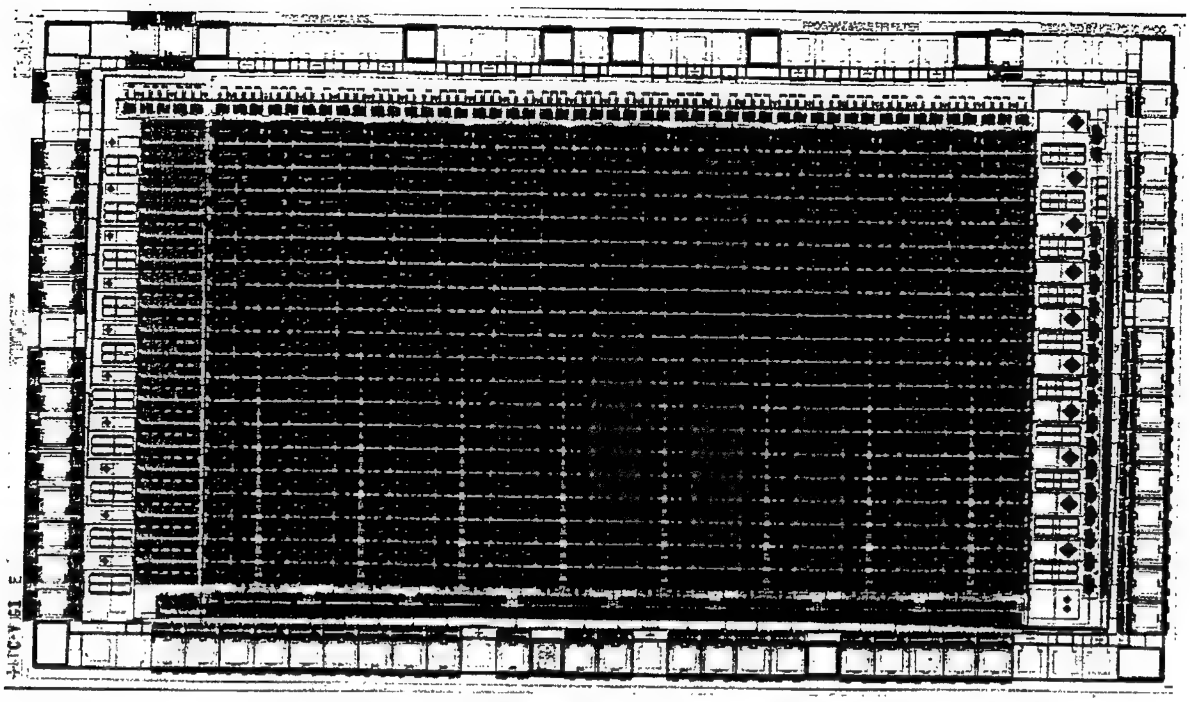
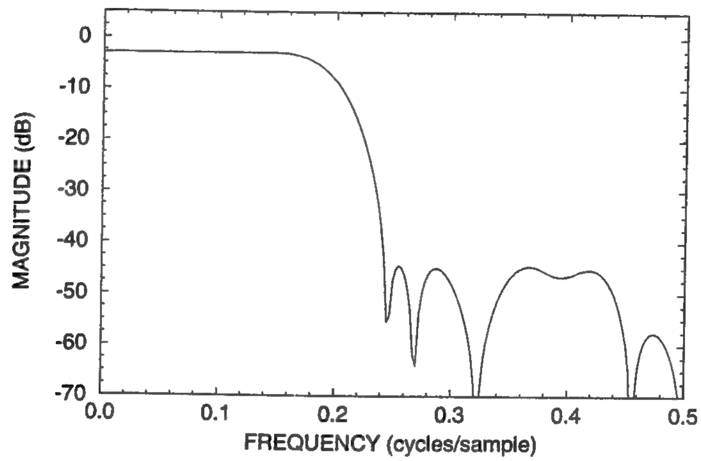


Figure 10: Photograph of the prototype chip.

can be extended to include more parallel programmable FIR filters for operations at the maximum data-rate while having filter taps with more than four non-zero CSD digits.

5 Design Examples

Three example filters have been designed to show the versatility of the proposed architecture: a 32-tap lowpass filter, a 16-tap lowpass filter, and a 32-tap bandpass filter. All three filter designs were constrained such that they could be implemented on the prototype chip described in Section 4 (i.e., at most 32 taps with two nonzero CSD digits per tap and a 16-bit shift range). With a larger filter core (i.e., more p-taps) more demanding filters could be implemented.

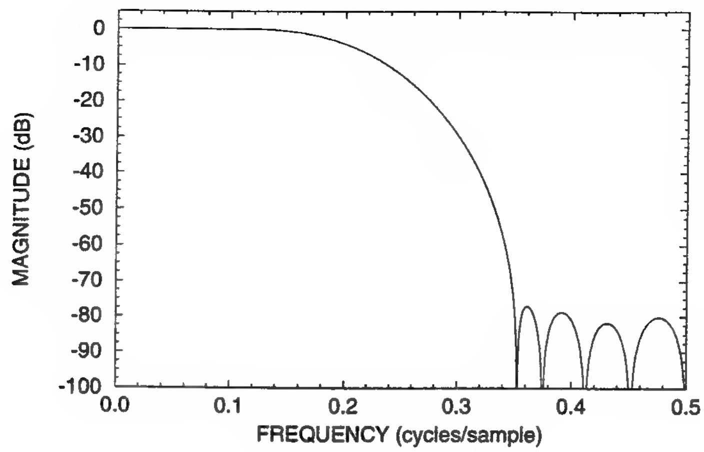


$h(0) = +2^{-11} + 2^{-14}$	$h(6) = -2^{-8} - 2^{-10}$	$h(12) = -2^{-4} + 2^{-7}$
$h(1) = -2^{-11}$	$h(7) = -2^{-6} + 2^{-9}$	$h(13) = -2^{-9} - 2^{-12}$
$h(2) = -2^{-9} - 2^{-11}$	$h(8) = -2^{-9} + 2^{-11}$	$h(14) = +2^{-3} + 2^{-6}$
$h(3) = -2^{-10} + 2^{-14}$	$h(9) = +2^{-5} - 2^{-7}$	$h(15) = +2^{-2} + 2^{-6}$
$h(4) = +2^{-8} + 2^{-11}$	$h(10) = +2^{-6} + 2^{-8}$	
$h(5) = +2^{-7} - 2^{-9}$	$h(11) = -2^{-5} + 2^{-7}$	

Figure 11: Frequency response and CSD coefficients for 32-tap FIR lowpass filter of Example 1.

Example 1: This example filter is a 32-tap lowpass filter with normalized passband and stopband edge frequencies of 0.15 and 0.25, respectively. The filter achieves a normalized stopband attenuation of 41.5 dB with a peak-to-peak passband ripple of 0.074 dB using only two nonzero CSD digits per tap. The coefficients for this filter and the corresponding frequency response are shown in Fig. 11. This filter requires a total of 32 p-taps when implemented on our prototype chip. Since this filter requires at most two coefficient digits per filter tap, the maximum data-rate achievable when implemented on our prototype chip is 180 MHz.

Example 2: This example filter is a 16-tap lowpass filter with normalized passband and stopband edge frequencies of 0.125 and 0.35, respectively. The filter achieves a stopband attenuation of 77.3 dB with a peak-to-peak passband ripple of 0.1 dB using three or four nonzero CSD digits per filter tap. The filter coefficients and the corresponding frequency



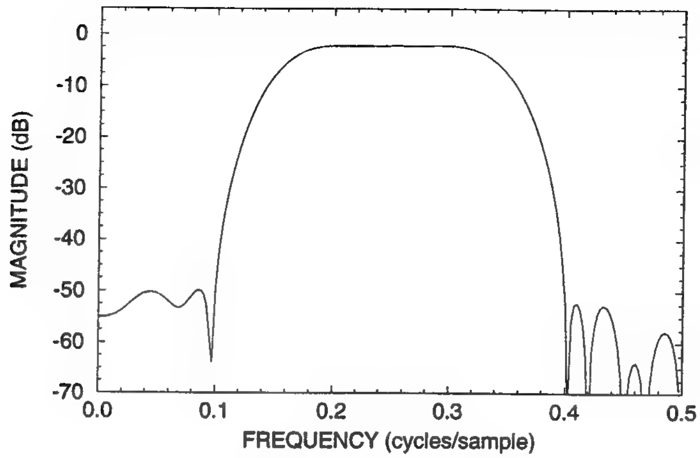
$h(0) = 2^{-11} - 2^{-14} + 2^{-16}$ $\quad h(4) = -2^{-4} + 2^{-7} - 2^{-10} - 2^{-15}$
$h(1) = 2^{-7} - 2^{-9} + 2^{-11} - 2^{-13}$ $\quad h(5) = -2^{-5} + 2^{-7} + 2^{-13} + 2^{-16}$
$h(2) = 2^{-6} - 2^{-9} - 2^{-11} + 2^{-13}$ $\quad h(6) = 2^{-2} - 2^{-4} - 2^{-6} + 2^{-9}$
$h(3) = -2^{-7} - 2^{-11} + 2^{-13} + 2^{-15}$ $\quad h(7) = 2^{-1} - 2^{-3} + 2^{-6} + 2^{-8}$

Figure 12: Frequency response and CSD coefficients for 16-tap FIR lowpass filter of Example 2.

response are shown in Fig. 12. When implemented on our prototype chip, this filter requires a total of 32 p-taps and therefore fully utilizes the hardware. The largest tap for this filter requires four nonzero digits (i.e., two p-taps) and therefore the maximum data-rate achievable by the prototype chip, for this filter, is 90 MHz

Example 3: This example filter is a 32-tap bandpass filter with the first stopband edge frequency of 0.1 (normalized), passband edge frequencies of 0.2 and 0.3, and the second stopband edge frequency of 0.4. The filter achieves normalized attenuation levels of 47.6 dB and 49.9 dB in the first and second stopbands, respectively, and a peak-to-peak passband ripple of 0.04 dB, while using only two nonzero CSD digits per filter tap. The coefficients for this filter, and the corresponding frequency response are shown in Fig. 13. This filter requires a total of 32 p-taps when implemented on our prototype chip. Like example 1, since the largest number of coefficient digits per filter tap is only two, the maximum data-rate achievable by our prototype chip, for this filter, is 180 MHz.

These three examples demonstrate the efficiency of our architecture. By contrast, a straightforward programmable FIR filter chip capable of implementing all three of these example filters with a uniform filter tap structure would require 32 taps with each tap having four nonzero digits. Thus, hardware for a total of 128 nonzero CSD digits would be required. In our prototype chip, however, we are able to implement all three filters using only 32 2-digit taps (p-taps), or a total of 64 nonzero digits—a savings of 50%.



$h(0) = -2^{-10} + 2^{-12}$	$h(6) = +2^{-8} - 2^{-10}$	$h(12) = +2^{-4} + 2^{-7}$
$h(1) = -2^{-11}$	$h(7) = -2^{-6} - 2^{-15}$	$h(13) = -2^{-3} - 2^{-7}$
$h(2) = -2^{-9} - 2^{-11}$	$h(8) = -2^{-5} + 2^{-8}$	$h(14) = -2^{-2} + 2^{-4}$
$h(3) = +2^{-8} + 2^{-10}$	$h(9) = +2^{-5} - 2^{-11}$	$h(15) = +2^{-2} - 2^{-5}$
$h(4) = +2^{-7} - 2^{-11}$	$h(10) = +2^{-6} + 2^{-9}$	
$h(5) = -2^{-8} - 2^{-13}$	$h(11) = +2^{-6} + 2^{-9}$	

Figure 13: Frequency response and CSD coefficients the 32-tap FIR bandpass filter of Example 3.

6 Conclusion

We have presented a new architecture for the implementation of the transposed FIR digital filter. We use a novel switchable unit-delay to allocate the optimal hardware resources to each filter tap. Moreover, a simple recoding of the coefficient values results in a simplification of the digit multiplication hardware. A prototype chip that can realize FIR filters with up to 32 linear-phase taps with 16-bit I/O has been implemented within a die size of 5.9 mm by 3.4 mm using 1.2-μm CMOS technology. The chip has been fabricated and tested to operate at data-rates up to 180 MHz.

While our new programmable structure is capable of implementing filters designed using existing algorithms for designing filters with Powers-of-Two coefficients, it will benefit from more specialized algorithms that can exploit our unique programmable-tap structure. That is, by taking advantage of our ability to use a small number of nonzero digits for many taps, we can expect to design significantly longer FIR filters than could be implemented with presently available CSD FIR approaches. A promising algorithm has been reported in [11], where the number of digits at each tap is variable and the optimization algorithm seeks to minimize the total number of coefficient digits for the entire filter. For the purpose of our filter, the optimization algorithm should minimize the pairs of coefficient digits at each tap while using as many filter taps as possible, subject to the available resources on a given filter chip.

References

[1] J. Evans, Y. Lim, and B. Liu, "A high speed programmable digital FIR filter," in *Proc. ICASSP-90*, vol. 2, pp. 969–971, Apr. 3-6, 1990.

[2] C. Golla, F. F. Nava, Cavallotti, A. Cremonesi, P. Piacentini, G. Casagrande, and G. Campardo, "A 30M samples/s programmable filter processor," in *ISSCC Dig. Tech. Papers*, pp. 116–117, Feb. 14-16, 1990.

[3] M. Hatamian and S. Rao, "A 100 MHz 40-tap programmable FIR filter chip," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 4, pp. 3053–3056, May 1-3, 1990.

[4] R. Hawley, T. Lin, and H. Samueli, "A silicon compiler for high-speed CMOS multirate FIR digital filters," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, pp. 1348–1351, May 10-13, 1992.

[5] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, Sept. 1961.

[6] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

[7] J. Laskowski and H. Samueli, "A 150-MHz 43-tap half-band FIR digital filter in 1.2-μm CMOS generated by silicon compiler," in *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 11.4/1–4, May 3-6, 1992.

[8] K.-Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "An efficient 175MHz programmable FIR digital filter," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 1, pp. 72–75, May 3-6, 1993.

[9] J. F. Kaiser, "Nonrecursive digital filter design using the I_o-sinh window function," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 20–23, Apr. 22-25, 1974.

[10] Y. Lim and S. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 583–591, June 1983.

[11] D. Li, J. Song, and Y. C. Lim, "A polynomial-time algorithm for designing digital filters with power-of-two coefficients," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 84–87, May 3-6, 1993.

[12] G. Reitwiesner, "Binary arithmetic," in *Advances in Computers*, vol. 1, pp. 232–351, New York: Academic Press, 1960.

[13] P. P. Vaidyanathan, *Multirate Systems and Filter Banks.* Englewood Cliffs, NJ: Prentice-Hall, 1993.

[14] T. G. Noll, "Carry-save architectures for high-speed digital signal processing," *J. VLSI Signal Processing*, vol. 3, pp. 121–140, June 1991.

[15] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, pp. 62–69, Feb. 1989.

[16] M. Abmramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design.* New York: Computer Science Press, 1990.

UNIVERSITY OF CALIFORNIA, LOS ANGELES

UCLA

BERKELEY • DAVIS • IRVINE • LOS ANGELES • RIVERSIDE • SAN DIEGO • SAN FRANCISCO

SANTA BARBARA • SANTA CRUZ

OFFICE OF INTELLECTUAL PROPERTY ADMINISTRATION
1400 UEBERROTH BUILDING
405 HILGARD AVENUE
LOS ANGELES, CALIFORNIA 90024-1406

April 13, 1993

National Science Foundation
1800 "G" Street N.W.
Washington, D.C. 20550

Office of Naval Research
California Institute of Technology
565 S. Wilson Ave.
Pasadena, CA 91106-3212

ATTENTION: Mr. John Chester, Esq.
Office of the Gen. Counsel

Mr. Clint Werner
Resident Representative

SUBJECT: **CONFIDENTIAL DISCLOSURE**
NSF Grant No. MIP-9201104,
Navy Grant No. N00014-91-J-1852
UCLA Invention Report No. LA93-008-01

Dear Mr. Chester/Mr. Werner:

We are in receipt of an invention disclosure entitled, "A Programmable Digital Signal Processor using Switchable Unit-Delays for Optimal Coefficient Allocation" which appears to have arisen under the subject funding agreements. Our office is in the process of reviewing the above referenced Invention Report to determine whether or not the University wishes to retain patent rights in the invention under Public Law 98-620. A copy of such Invention Report is enclosed for your records. I shall report the University's election to you as soon as a determination has been made. Please let me know which of your agencies wishes to take lead status.

In the meantime, we ask that no public distribution or publication be made of the Invention Report.

Sincerely,

ORIGINAL SIGNED BY
JAMES C. SMART

James C. Smart, Ph.D.
Technology Transfer Officer

Enclosure: Invention Report No. LA93-008-01

cc w/o encl: Dr. Alan N. Willson, Jr.
Mr. Todd R. Patrick
Mr. Hardy Dhillon

UNIVERSITY OF CALIFORNIA, LOS ANGELES

UCLA

BERKELEY · DAVIS · IRVINE · LOS ANGELES · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

To: Jim Smart
Technology Transfer Officer

From: Dr. Alan N. Willson, Jr.
Professor

Date: February 9, 1993

Subject: Invention Report

1. **A Programmable Digital Signal Processor using Switchable Unit-Delays for Optimal Coefficient Allocation.**

2 A novel *switchable unit-delay* has been developed for the efficient implementation of programmable digital FIR filters and correlators using the canonical signed digit (CSD) approach [1]. Our design enables high-speed processing while avoiding the severe hardware inefficiency that would result from straightforward programmable tap implementations that were reported previously [2,3,4]. (In a straightforward implementation many tap "multipliers" would significantly waste valuable computational resources since all taps of a *programmable* structure would need to accommodate "difficult" coefficient values, while for any specific transfer function, most taps would not require such extreme capabilities.) The switchable unit-delay not only allows the programming of the number of taps and the specific tap-coefficient values, it provides the capability for programming the optimal allocation of hardware resources to each tap. Thus the computational resources that otherwise might have been wasted are made available to further increase the precision in any tap's coefficient representation, or for use in implementing a larger number of taps. This capability is critical for the feasible VLSI implementation of long FIR filters and correlators used in high-end digital signal processing applications. Our prototype chip demonstrates the ability to implement a broad spectrum of linear-phase FIR filters employing up to 32 taps with 16-bit input and output data and operating at data rates as high as 175MHz (simulated) in a die size of 5.9mm by 3.4mm using 1.2μm CMOS technology.

3. This work was supported by the Office of Naval Research under Grant N00014-91-J-1852 and by the National Science Foundation under Grant MIP-9201104. The Principal investigator was Dr. Alan N. Willson, Jr.

4. No proprietary materials or computer software was used in this work.

5. The invention was first conceived of in January 1992.

6. The invention was first tested in simulation in November 1992.
7. The invention has been orally described to the following people:
 David Thornhill and Chuck McGown from TRW on 15 Oct 1992
 In addition, written copies have been distributed to:
 David Thornhill, Greg Shreve, Robert Harnden and Jeff Mullin of TRW in Feb 1992.
 Dr. Tran Thong of Tektrnoix on 26 May 1992.
 Office of Naval Research in January 1993 progress report.
 Tim Knerr, Dan Azaren and John Schimm of TRW on 5 Feb. 1993.

8. This invention will be presented at the IEEE International Symposium on Circuits and Systems (ISCAS'93) to be held in Chicago, May 3-6, 1993. A written copy will be included in the Symposium Proceedings.

9. References:

1. Kei-Yong Khoo, Alan Kwentus, and Alan N. Willson, Jr., "An efficient 175MHz programmable FIR digital filter," *Proc. IEEE Int. Symp. Circuits and Systems*, May 3-6, 1993.
2. J.B. Evans, Y.C. Lim, and B. Liu, "A high speed programmable digital FIR filter," *Proc. ICASSP-90*, vol. 2, pp. 969-71, April 3-6, 1990.
3. M. Hatamian and S.K. Rao, "A 100 MHz 40-tap programmable FIR filter chip," *Proc. Int. Symp. Circuits and Systems*, vol. 4, pp. 3053-6, May 1-3, 1990.
4. C. Golla, F. Nava, F. Cavallotti, A. Cremonesi, P. Piacentini, G. Casagrande, and G. Campardo, "A 30M samples/s programmable filter processor," *Proc. IEEE Int. Solid-State Circuits Conf.*, pp. 116-117, 1990.

10. Companies interested: TRW, Bell Labs, Harris Semiconductor, SGS-Thomson Microelectronics.
11. Contributors to the Invention

Name: Kei-Yong Khoo
Department: Electrical Engineering
Campus: Los Angeles
Address: 56-122 Engineering IV, # N38, UCLA, Los Angeles, CA 90024
Telephone: 310-206-2573

Name: Alan Kwentus
Department: Electrical Engineering
Campus: Los Angeles
Address: 56-122 Engineering IV, # N39, UCLA, Los Angeles, CA 90024
Telephone: 310-206-2573

Name: Dr. Alan N. Willson, Jr.
Department: Electrical Engineering
Campus: Los Angeles
Address: 66-147H Engineering IV, UCLA, Los Angeles, CA 90024
Telephone: 310-825-7400

12. Witnesses

James Chang 2/9/93
James CHANG

Avanindra 2/9/93
Avanindra. MADISETTI

Automated Programming of Digital Filters for Parallel Processing Implementation

Michael J. Werter, *Member, IEEE*, and Alan N. Willson, Jr., *Fellow, IEEE*

Abstract— **A computer algorithm is described that automatically writes optimal programs for the implementation of arbitrary digital filter structures on parallel processors. The algorithm has been adapted particularly for programming a DSP chip with multiple processors arranged in a ring-type topology. The algorithm starts from a netlist describing a desired digital filter structure. The algorithm's output is a set of programs for the parallel processors which causes them to implement the given digital filter.**

I. Introduction

This paper describes a computer algorithm that automatically writes programs for the implementation of digital filters on parallel processors. It has been used for implementing many common filter structures on a new digital signal processing (DSP) chip [1], [2]. The programs are optimal; that is, they use the minimum number of program steps per data sample to implement a given arbitrary digital filter structure. The algorithm's "input" is a netlist describing the desired digital filter structure, which is used to define a shift-invariant data-flow graph: a directed graph in which all operations (additions, multiplications, and time delays) are specified at the nodes, and in which the branches are directed paths specifying the flow of data between nodes [3]–[5]. The algorithm first optimizes this flow graph to achieve the best performance from the parallel processors when implementing the given filter structure. It next calculates a time schedule for the flow graph's arithmetical operations and then distributes these operations over the multiple processors, taking into account all the restrictions which appear due to the topology and the processors' architecture. The algorithm's "output" is a set of programs for the parallel processors which causes them to implement the given digital filter.

For the reader's convenience, some properties of the programmable digital filter IC presented in [1] are briefly reviewed in Section II. In Section III the computer algorithm is described. In Section IV we compare our algorithm with other scheduling algorithms, and we summarize the main results in Section V.

Manuscript received September 1, 1992; revised June 15, 1993. This paper was recommended by the Associate Editor Y. C. Lim. This work was supported by the Office of Naval Research under Grant N00014-91-J-1852 and by a grant from the State of California and TRW, through the California MICRO program.

The authors are with the Electrical Engineering Department at the University of California, Los Angeles, CA.

IEEE Log Number 9400240.

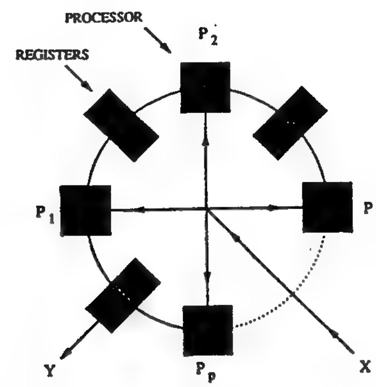


Fig. 1. Ring-structured processor topology.

II. A Ring-Structured Topology for Digital Filtering

In [1] a DSP chip is described that contains multiple processors placed in a ring-structured topology on a single integrated circuit (Fig. 1). Due to this ring structure the communication between processors is restricted to neighboring processors only. For the implementation of many popular digital filter structures this restriction produces no disadvantage over more complex communication schemes; it has been shown in [1] that the ring-structured parallel processor system can implement filters using the minimum possible number of sequential arithmetic operations per data sample.

Since the intended application of the DSP chip is real-time digital filtering, the processors need only be able to perform the five instructions: add, subtract, multiply, move (register to register), and nop (no operation). The ALU consists of a hardware multiplier, a RAM to store multiplier coefficients and an adder/subtractor, as shown in Fig. 2. The ALU is pipelined so that the multiplier will execute in one clock cycle. This way, it can perform an addition and a multiplication simultaneously. Since it happens that most digital filters perform an addition immediately following a multiplication, this ALU architecture makes it possible to perform both functions in "essentially" one instruction step.

III. Computer Algorithm

In a general-purpose computing context the major difficulty with most parallel architectures is specifying how to program them. However, since digital filters require no conditional branching it is possible to write a computer algorithm (a task partitioner) that analyzes a structural description of a given filter and writes optimal programs for parallel processing. We have developed such an algorithm. Its flow graph is shown in

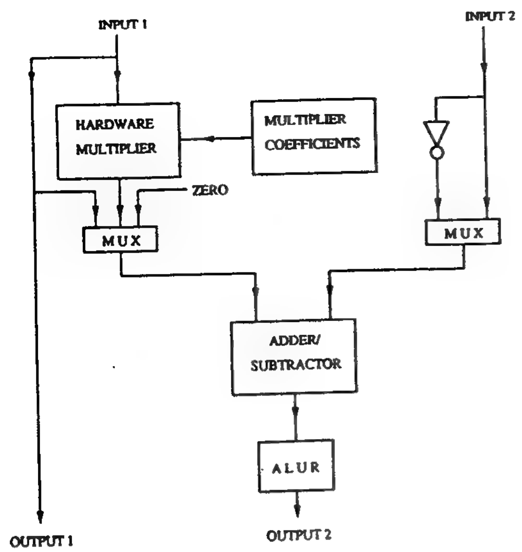


Fig. 2. ALU architecture.

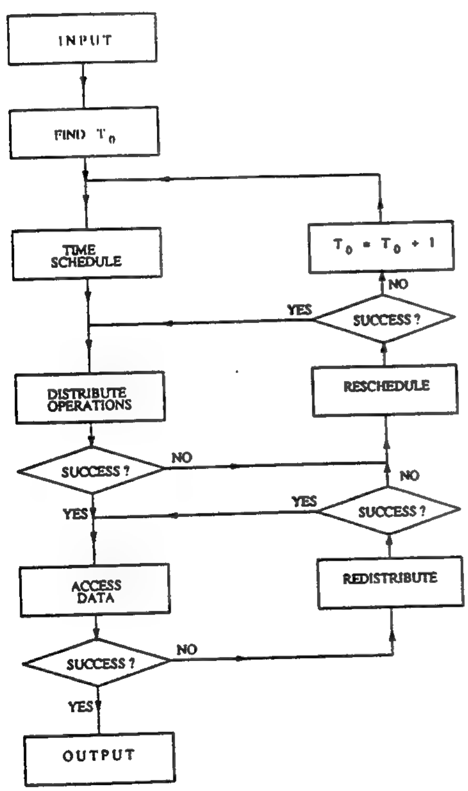


Fig. 3. Flow graph of computer algorithm.

Fig. 3. In this section, the algorithm will be explained with the aid of an example.

Section III will show that the filter programs written by the algorithm are optimal; that is, they use the minimum number of program steps per data sample to implement a given arbitrary digital filter structure. The algorithm calculates the optimum sampling period T_o (Section III–B), it searches for an optimum schedule (Section III–C) and optimum distribution of the operations over the processors (Section III–D). If (and only if) it is not possible to implement the given digital filter at the optimum sampling period, the algorithm increases the sampling period by one time unit (Section III–F).

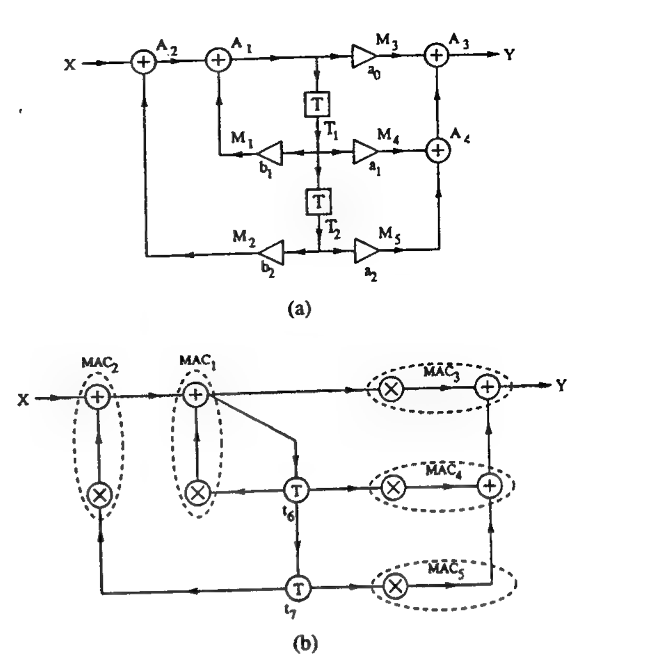


Fig. 4. Second-order direct form II filter. (a) Filter structure, (b) shift-invariant data-flow graph.

A. Input

The computer algorithm starts with a netlist describing a desired digital filter structure. That is, the algorithm's input data specify a shift-invariant data-flow graph by stating how each node k (representing an addition, multiplication or time delay) is connected with other nodes of the flow graph. To avoid ambiguities, we assume that each adder has two ingoing branches and one outgoing branch. We also assume that any desired filter is "realizable" [6], [7], i.e., that it contains no delay-free directed loop. In addition, the filter is assumed to be "proper" [8] in the sense that there is a directed path from the input to every node and a directed path to the output from every node in the data-flow graph. In other words, all parts of a proper filter affect the input/output behavior.

As an example, Fig. 4(a) shows the topology of a second-order direct form II digital filter and the corresponding data-flow graph is shown in Fig. 4(b). The netlist input file of this example is shown in Appendix C.

Our algorithm first combines each multiplication with a subsequent addition into a two-step Multiply-Accumulate (MAC) instruction. If no addition follows a multiplier, "zero" will be added to it during the accumulate stage[1]. The MAC instructions are called *supernodes* in the data-flow graph and they are depicted by dashed ellipses in Fig. 4(b).

B. Optimum sampling period T_o

Using a well-known technique of Renfors and Neuvo [8], [9], our algorithm calculates the theoretical minimum sampling

[1] If a time-delay element is located between a multiplier and an adder, then the sequence of the time-delay and the multiplier will be reversed, so that the multiplier can be integrated with the adder in a MAC instruction.

period[2] T_{min} that would be possible for any custom parallel implementation of the specified filter structure assuming that an unlimited number of processors are available. Therefore it searches for all directed loops in the data-flow graph. For every directed loop l, it counts the number of time-delay nodes N_l and it calculates the arithmetic loop delay D_l, which equals the total processing time consumed by the arithmetical operations in the loop. The minimum sampling period T_{min} is calculated by

$$T_{min} = \max_{l}(D_l/N_l) \tag{1}$$

where the maximum is taken over all directed loops in the flow graph. A directed loop in which this maximum is reached is called a critical loop.

An alternative way to compute the minimum sampling period (iteration period bound) is based on the longest-path matrices and their multiplication [10]. An advantage of that algorithm is that it has a polynomial complexity, while the search for all possible directed loops in a data-flow graph can grow as a factorial function of the number of time-delay nodes, as discussed in Appendix A. The program can easily be modified to support this alternative approach.

The second-order filter example of Fig. 4 has two recursive loops. The minimum sampling period T_{min}, calculated with the Renfors and Neuvo algorithm [8], is found from the loop containing M_1, A_1 and T_1:

$$T_{min} = T_M + T_A$$

where T_M and T_A denote the time needed for a multiplication and an addition, respectively.

Since the implementation of the desired digital filter structure must be accomplished on a limited number of processors P, our algorithm next calculates T_P, the minimum total computation time per processor. The average computation time of a MAC instruction[3] equals T_M. The computation time of an add, subtract or move instruction equals T_A, so the minimum total computation time per processor T_P for an implementation of the desired digital filter structure on a limited number of processors P equals

$$T_P = \frac{K_1 \cdot T_M + K_2 \cdot T_A}{P} \tag{2}$$

where K_1 is the total number of supernodes and K_2 is the total number of other nodes in the data-flow graph which are not time-delay nodes.

The optimum sampling period for the implementation of the flow graph on a multiple processor system with P processors can now be calculated by

$$T_o = \max(T_{min}, T_P). \tag{3}$$

From (2) and (3) we can calculate P_{min}, the minimum number of processors that is needed to implement a digital filter at the

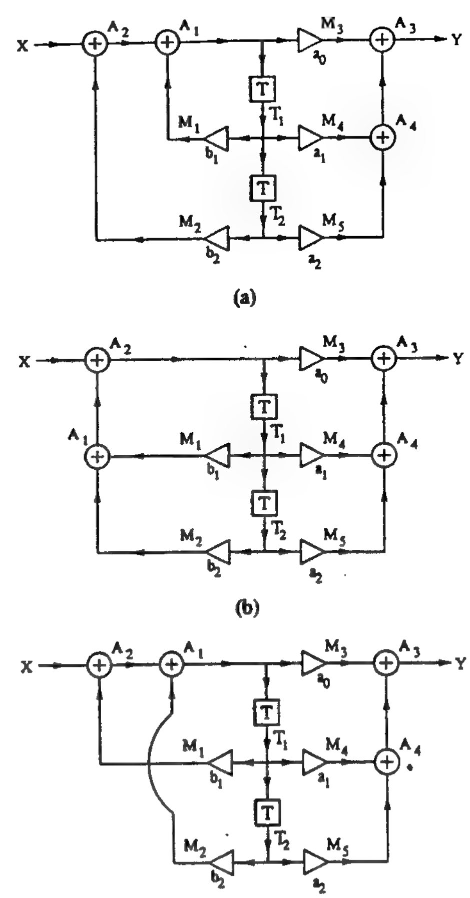


Fig. 5. Three topologies of second-order direct form II filter.

theoretical minimum sampling period T_{min}:

$$P_{min} = \left\lceil \frac{K_1 \cdot T_M + K_2 \cdot T_A}{T_{min}} \right\rceil. \tag{4}$$

From (4) we conclude that the minimum number of processors needed to execute the five MAC instructions of the second-order filter example of Fig. 4 on a system with $T_A = T_M$ = one step at the theoretical minimum sampling period T_{min} = two-steps is $P_{min} = 3$ processors.

In Fig. 4(a) we see four two-input adders. Adders A_1 and A_2 can, however, be considered as one three-input adder with ingoing branches from input x and multipliers M_1 and M_2. This three-input adder could be implemented in three different ways, as shown in the three Fig. 5 structures. As is well known, all three implementations produce the same three-input sum if two's complement arithmetic is used for quantization and overflow correction. Of the three filters shown, only the first has the minimum sampling period T_{min} = two-steps; the other two have loops with two adders, one multiplier and one time delay so that T_{min} = three steps. This shows that it is important to optimally sequence ingoing branches of a multiple-input adder. For this reason our algorithm detects all multiple-input adders in the desired filter structure and provides the user the option of searching for the optimum adder sequence to minimize the filter's sampling period. If this option is used the algorithm recursively splits each N-input

[2] The minimum sampling period T_{min} has been called the *iteration period bound* in [3], [10], and its reciprocal value is, of course, the *maximum sampling rate* [8].

[3] Our algorithm can handle pipelined multipliers in which a multiplication would be executed in multiple stages.

adder with $N > 2$ into an M-input adder and an $(N - M)$-input adder, where $1 \leq M \leq N/2$. (A one-input adder is simply a branch in the data-flow graph.) There are $\binom{N}{M}$ different combinations into which the N-input adder can be split, yielding an M-input adder and an $(N - M)$-input adder, where $\binom{N}{M} = \frac{N!}{M!(N-M)!}$. Let $S(N)$ be the total number of different combinations into which an N-input adder can be split, then $S(1) = 1$, and

$$S(N) = \frac{1}{2}\sum_{M=1}^{N-1}\binom{N}{M}S(M)S(N-M) = (2N-3)!! \qquad \text{for } N \geq 2$$

where $i!! = i(i-2)(i-4)\ldots 5 \cdot 3 \cdot 1$ for i odd. A proof of this result is given in Appendix B. The total number of combinations increases rapidly with N, therefore it is not practical to check all combinations for adders having many ingoing branches. In most filter structures, however, multiple-input adders with many ingoing branches are rare (perhaps the most noteworthy exception being the direct-form FIR structure). Thus it is, in fact, usually feasible to employ our algorithm's option to search for the filter topology with optimally-sequenced adders that yields the lowest minimum sampling period T_o.

C. Time schedule

Having found T_o the flow graph of Fig. 3 shows that the next task is to determine the time schedule. The earliest time $T(k)$ at which the operation at node k can be started is found from a maximal distance spanning tree [11], which is a tree of the data-flow graph containing all of the flow graph's nodes, having the property that there is a directed path from the input to each node k, such that the sum of all processing times in such a path is maximal. The individual nodes in a supernode of the data-flow graph cannot be separated in the maximal distance spanning tree since they represent the addition and multiplication of a single MAC instruction. Therefore, a branch within a supernode is always a part of the maximal distance spanning tree. As discussed in [8], the processing time of a time-delay node equals $-T_o$ (a negative value!), which causes the total processing time of a critical loop to be zero, while the processing time of all directed noncritical loops have negative values. The latter implies that there is some (positive) slack time between the time that the execution of an operation is completed and the time that the result of this operation is needed for further processing. These slack times have been called "shimming delays" [7], and they can be depicted as (positive) shimming-delay blocks in some of the branches of the data-flow graph. After insertion of all shimming-delay elements into the data-flow graph the total processing time of *each* loop (directed or nondirected) equals zero, where in a nondirected loop the sign of the processing time of a node operation or a shimming delay is reversed if its direction is opposite to the loop's reference direction in the data-flow graph. A maximal distance spanning tree is found using an algorithm similar to the Bellman-Ford method [12].

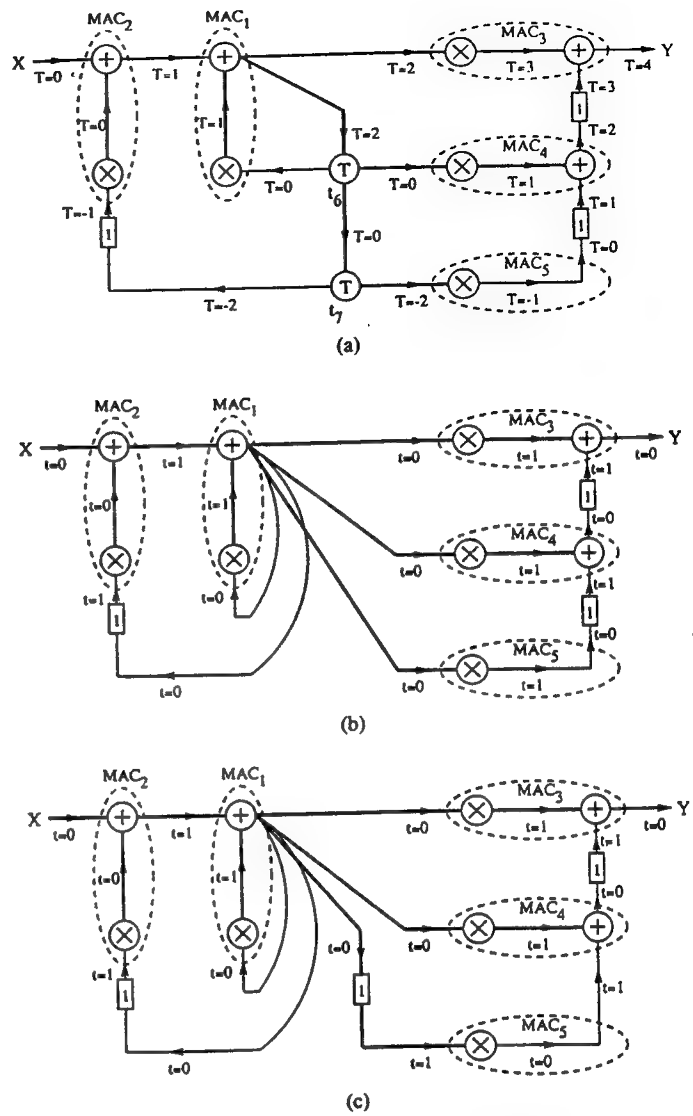


Fig. 6. Second-order direct form II filter. (a) Original data flow graph, (b) data-flow graph after deleting of time-delay nodes, (c) data-flow graph after rescheduling.

In Fig. 6(a), a maximal distance spanning tree for the data-flow graph of the Fig. 4 filter example is shown by thick-lined branches. From the maximal distance spanning tree we calculate the earliest time at which the operation at (super) node k can be started. The results are shown in Fig. 6(a) assuming a new input data sample is available at time $T = 0$. The algorithm next finds the shimming delays, which are depicted in Fig. 6(a) by rectangular boxes. Notice that the critical loop contains indeed no shimming delay.

Since all operations are executed periodically, and since an instruction writes data to the same register as that of the previous sample period, all shimming delays should have values less than the sampling period T_o. If a shimming delay's value equals or exceeds T_o a newly produced data sample would be written over old data before it has been used for further computations. This problem can be prevented by employing additional move instructions which copy the old data (moving it to another register) before the new data is

produced. Alternatively, we can sometimes avoid this problem by rescheduling the operations, or by unfolding the data-flow graph [14]. In the second-order filter example the problem does not arise since all shimming delays have values which are less than T_o.

Since all operations are executed periodically with the sampling period T_o we next modify the time schedule by specifying its values modulo T_o: if the time $T(k)$, which is the sum of the processing times from the input node to the node k, according to the maximal distance spanning tree, equals

$$T(k) = m \cdot T_o + t(k), \quad \text{with } m = \text{integer}, \quad 0 \leq t(k) < T_o$$

then operation k will be scheduled to start at step $t(k)$ in the program. Notice that the time-delay nodes in the data-flow graph have no effect on the value of $t(k)$; consequently they are now removed ("short-circuited"). The time schedule for the second-order filter example is shown in Fig. 6(b).

D. Operation distribution

According to the Fig. 3 flow graph, we must next determine how the operations will be distributed over the processors, and then check for the accessibility of data. If we find that it is not possible to appropriately distribute the operations over the processors the operations will be rescheduled; that is, one of the operations will be selected to start at a different time (a different step in the program). In the data-flow graph a rescheduling can be visualized as a pushing of shimming-delay elements through the nodes. The algorithm also adds shimming delays at the filter's input and output node, which are used in the rescheduling process. Except for a pipeline delay, these additional shimming delays do not change the filter operation; the new filter performs the same sequence of multiplications and additions, and thus has the same behavior with respect to quantization errors, as the filter without the additional shimming delays. Our algorithm checks which operations can be rescheduled and it reschedules one of these operations in searching for a solution. It also keeps track of how much each operation is shifted from the original time schedule, to prevent duplication of rescheduling operations. The rescheduling is repeated every time the program fails to (re)distribute operations over the processors, until all possible time schedules have been checked.

In the Fig. 6(b) data flow graph of the second-order filter example we see that there are four MAC instructions which use the result of MAC_1 at time $t = 0$; that is, immediately after it has been produced. At that time the MAC_1 result will be available at the ALUR of the processor where it has been produced. Since the ALUR of each processor in the ring structure is only accessible by its own processor and by its two neighbors, we can execute only three instructions at time $t = 0$ which use the MAC_1 result. Therefore we have to reschedule one of the operations, MAC_5 for example, so that it starts at $t = 1$. The rescheduling of MAC_5 does not change the sequence in which the operations are executed since there was a shimming delay of one-step between MAC_5 and adder A_4. After this rescheduling the second-order direct form II filter example can be executed on three processors, which is the minimum number of processors needed to implement this filter, at the optimum sampling period according to (4). The data-flow graph after rescheduling is shown in Fig. 6(c).

The rescheduling described in this section may seem equivalent to the retiming technique used in [13], which redistributes time-delay elements over a filter structure and so creates new time schedules. Our algorithm, however, does not redistribute time-delay nodes (which have been deleted from the data-flow graph after a maximal distance spanning tree was found), but it redistributes the shimming-delay elements over the flow graph. And while the retiming technique can improve the sampling period of an implementation of a digital filter but cannot guarantee a schedule to be rate-optimal [14], all our implementations operate at the optimum sampling period T_o.

The initial distribution of operations over the parallel processors assigns each operation to the processor with lowest index that is free during the complete time it takes to execute this operation. Notice that the operation of checking for a free processor is performed modulo the sampling period T_o, since all operations are executed periodically. Each redistribution assigns an operation to the next available processor. In this way all possible distributions of the operations over the parallel processors can be tested.

E. Data accessibility

After the operations are distributed over the processors, the computer algorithm checks whether all data can be made accessible to all processors that need it. Therefore, for each data sample, a list of processors needing it is formed. If a processor needs data that has just been produced by itself or one of its neighbors, this processor can be removed from the list, since the data can be accessed via an ALUR. For all processors that remain on the list, the program checks whether "in-between processors" can move the data from the processor where it was produced to the one where it is needed.

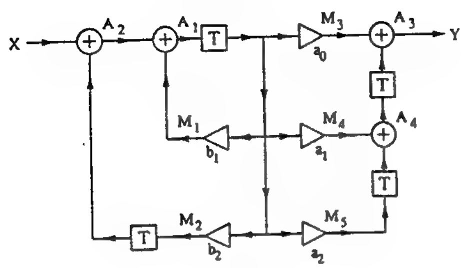
F. Output

The "output" of the computer algorithm is a set of programs for the parallel processors which causes them to implement the given filter structure.

It is easy to show how a parallel processor DSP chip containing as few as three processors can implement the general second-order direct-form II filter of Fig. 4(a). The parallel (two-step) programs that our algorithm finds are given in Fig. 7(a). The filter that the Fig. 7(a) programs implement is shown in Fig. 7(b). This Fig. 7(b) filter structure can easily be derived from the Fig. 6(c) data-flow graph by inserting time-delay nodes in every branch where $t = 0$ (including branches within supernodes) except for the branches which leave from the input x, or those going to the output y. Notice that the Fig. 7(b) structure actually implements the transfer function $z^{-1}H(z)$, which differs from the specified $H(z)$ by one pipeline delay. In general, the algorithm is capable of taking advantage of additional pipeline delays to achieve efficient implementations of the specified filter, so being aware of, and planning for such modifications need not be a concern to the user.

	P_1	P_2	P_3	$P_4 \ldots P_P$
Step 1	$M_2 + X \rightarrow A_2$	$A'_4 \leftarrow A_4$	$A_5 \leftarrow M_5 + 0$	NOP
	$M_3 = a_0 \times A_1$	$M_1 = b_1 \times A_1$	$M_4 = a_1 \times A_1$	NOP
Step 2	$Y \leftarrow M_3 + A'_4$	$A_1 \leftrightarrow M_1 + A_2$	$A_4 \leftarrow M_4 + A_5$	NOP
	$M_2 = b_2 \times A_1$		$M_5 = a_2 \times A_1$	NOP

(a)



(b)

Fig. 7. Implementation of second-order direct form II filter. (a) Programs, (b) implemented filter structure.

An examination of the programs will demonstrate how data flows, from the input x to the output register y, as the processors communicate with each other by means of their adjacent shared register blocks and ALUR registers.

We use the arrows shown in the programs to indicate which of the two adjacent blocks the output data is directed to. Thus, we indicate a clockwise-directed output by a right-pointing arrow, and a counterclockwise-directed output by a left-pointing arrow. The two-way directed arrow in step 2 at processor P_2 in Fig. 7(a) shows that data A_1 will be stored in the register blocks at *both* sides of processor P_2.

To speed up the task partitioner's distribution of operations over the processors and simultaneously to reduce the communication between processors, our algorithm has the option to add the additional constraint that an operation may only be assigned to that processor where its input data are created, or one of the adjacent processors.

While we have been able to implement all practical examples of digital filter structures at the optimum sampling period T_o on the ring of processors, it is possible to create contrived data-flow graphs for which an implementation on P processors operating at a sampling period T_o does not exist [15]. If this occurs the sampling period T_o will be increased by one time unit, and the algorithm will then continue with the initial time scheduling, as shown in Fig. 3.

IV. Comparison With Other Scheduling Algorithms

The computer algorithm presented in this paper has some similarities with the range-chart-guided iterative data-flow graph scheduling of [15]–[18]. In [15] the following scheduling methods have been compared with each other:

1) Single iteration methods [20], [21];
2) Direct blocking methods [4], [18];
3) Fixed rate methods based on:
   a. Maximal distance spanning tree [8], [9];
   b. Optimum unfolding [14], [22], [23];
   c. Cyclo-static scheduling [3]–[5].

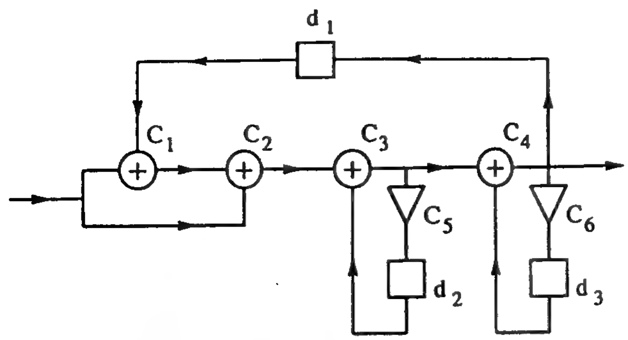
In this section, we shall therefore compare our algorithm with the algorithm of [15].

All algorithms start with a description of the desired digital filter by a data-flow graph. Our algorithm is the only one that first combines each multiplier with a subsequent adder in a two-step MAC-instruction; this, of course, is dictated by the advantage of implementing a MAC operation as a single instruction on our hardware, which reduces the total number of instructions significantly. The number of instructions in an FIR filter program, for example, is reduced by 50 percent [1].

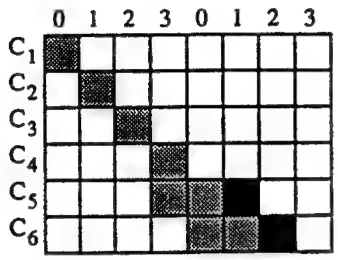
Similar to most scheduling algorithms, we first calculate the minimum sampling period, assuming that an unlimited number of processors are available. Our algorithm is the only one that has the option to automatically change the topology of multiple-input adders and so improve the minimum sampling period. Unlike all scheduling algorithms with the exception of the range-chart-guided scheduling, we also calculate the optimum sampling period assuming that a limited number of processors are available. We then calculate the amount of time over which operations can be rescheduled. We use the maximal distance spanning tree to find the earliest time at which each operation can be started, so our "reference node" [15] is the input node. Since all operations are executed periodically we next delete the time-delay nodes from the data-flow graph, taking care that shimming delays do not exceed or equal the sampling period T_o. Therefore our scheduling range is limited to $0 \leq t < T_o$. The range-chart-guided scheduling does not necessarily place a T_o limit on a program's schedule. In fact, it appears that the implementation of the second-order direct form II filter shown in Fig. 13(b) of [15] has a new c_8 data value calculated before the old c_8 value has been used in the addition that forms c_6. Therefore, if the programs would write data to the same register each sampling period, the processor P_4 at time $t = 0$ performs the incorrect addition $c_6(i) = c_7(i) + c_8(i+1)$.

The time schedule found with the range-chart-guided scheduling algorithm does *not* necessarily produce the minimum number of levels. An example in which the algorithm produces more than the minimum required number of levels is shown in Fig. 8. Fig. 8(a) shows a circuit with four additions (one-step), two multiplications (two-steps), and three time delays. The minimum sampling period is $T_{min} = 4$ steps, which is dictated by the loop c_1, c_2, c_3, c_4, d_1. If we select c_1 as our reference node the scheduling-range chart will become as shown in Fig. 8(b). Following the scheduling algorithm presented in [15] the sequence in which the operations are to be scheduled is: 1, 2, 3, 4, 5, 6. The final equivalence class is shown in Fig. 8(c). When operation 5 is placed at the upper fixed limit, we need a third level to place operation 6. The algorithm therefore does not find the optimum solution which has only 2 levels and can be implemented on two processors, as shown in Fig. 8(d).

Even if the number of levels found in the range-chart-guided scheduling algorithm is minimal, this program does *not* guarantee an implementation on the minimum number of processors, as shown in the following example. Consider the assignment of operations in Fig. 9(a). In the range-chart-guided scheduling algorithm the operations will be assigned



(a)



(b)

	6		
5	5	6	
1	2	3	4

(c)

5	6	6	5
1	2	3	4

(d)

Fig. 8 Example with no optimum range-chart-guided scheduling. (a) Filter structure, (b) scheduling-range chart, (c) scheduling accoding to [15], (d) optimum scheduling.

t =	1	2	3	4	5	6	7	8	9	0
P_2	5	5	5	5	5	6	6	7	7	7
P_1	1	2	2	3	3	3	4	4	4	4

(a)

P_3						6	6			
P_2	1	2	2	3	3	3		7	7	7
P_1	5	5	5	5	5		4	4	4	4

(b)

Fig. 9. Example with no optimum range-chart-guided distribution. (a) Schedule of operations to be distributed, (b) processor distribution according to [15].

to processors in the sequence: 5 to P1, 4 to P1, 3 to P2, 7 to P2, 2 to P2. At this moment, operation 6 must be assigned to a processor. However, since processor P1 is used by operation 4 at time $t = 7$, and since processor P2 is used by operation 3 at time $t = 6$, we have to use a third processor to execute operation 6, as shown in Fig. 9(b). This distribution of operations over the processors is not optimal, since we can assign the original schedule immediately to 2 processors.

One of the differences between our algorithm and the range-chart-guided scheduling algorithm is that the latter does not try to reschedule operations if the number of levels exceeds the number of processors, nor does it try to redistribute these operations if the assignment of operations to the processors is unsuccessful.

Finally, none of the other scheduling algorithms seems to have employed a scheduling of operations to processors where data communication is restricted due to processor topology constraints.

V. Conclusions

In this paper, a computer algorithm has been described that automatically writes optimal programs for parallel processors. The algorithm has been adapted particularly for programming a DSP chip with multiple processors arranged in a ring-type topology, but it can easily be modified for other multiprocessor digital filter chips. The algorithm can check all possible timing schedules and all possible distributions of the operations over the parallel processors, taking into account the constraints imposed by the multiprocessor topology and the processors' architecture. It searches iteratively for a set of programs that implements the given digital filter at the optimum sampling period on a limited number of processors.

We have proved with this computer algorithm that, among others, the following common filter structures can be implemented to execute in the optimal manner on a single chip which contains five processors in a ring-type topology [1]: cascades of second-order direct-form II filters (one program step per second-order section, for cascades of two or more filter sections), arbitrary FIR filter (one program step per five filter taps), a 10-th order Gray-Markel lattice filter (five-step program), a general second-order state-space filter (three-step program), and a fifth-order wave digital filter (nine-step program).

Since FIR filters and cascades of second-order direct-form II filters are the most common digital filter structures, we have created a library for the programs that implement these types of filters with arbitrary order. At the start of our algorithm, the user has the option to directly call programs from the library, and after completion of the scheduling of the operation over the processors the user can add newly-found programs to this library.

While the algorithm exhibits a worst-case running time which increases rapidly with the number of instructions in the data flow graph, for all practical examples the running time was quite acceptable. Table I in Appendix A compares the number of computations for the calculation of the minimum sampling period of a "worst-case " example filter using the Renfors-Neuvo method (used in our algorithm) and using a polynomial-time algorithm. Notice that, it is only when the order of a filter section equals or exceeds seven that the polynomial-time algorithm outperforms our algorithm. This fact, along with the fact that worst-case examples are hardly ever encountered, accounts for the quite acceptable performance of our algorithm, even though in principle, one could expect to sometimes encounter unreasonably long computation times.

TABLE I
TOTAL NUMBER OF COMPUTATIONS FOR THE CALCULATION OF THE MINIMUM SAMPLING PERIOD T_{min} OF AN N-TH ORDER GRAY-MARKEL LATTICE DIGITAL FILTER WITH THE RENFORS-NEUVO METHOD (M_1) AND WITH THE POLYNOMIAL-TIME ALGORITHM (M_2).

N	M_1	M_2
1	5	9
2	15	72
3	43	272
4	136	710
5	534	1,497
6	2,629	2,756
7	15,812	4,614
8	112,504	7,205
9	921,598	10,670
10	8,525,229	15,151

VI. APPENDIX A

In [10] it has been shown that in a data-flow graph in which there is a directed path between every pair of time-delay nodes (e.g., a Gray-Markel lattice digital filter [19]) the total number of loops equals

$$L = \sum_{k=1}^{N} \frac{N!}{k \cdot (N-k)!} \tag{5}$$

where N is the total number of time-delay nodes in the data-flow graph. An algorithm that finds the minimum sampling period T_{min} of a Gray-Markel lattice filter using (1) must perform a total of $M_1 = G \cdot L$ computations, where G is the average number of arithmetical nodes in the loops of the data-flow graph. In the Gray-Markel lattice filter $G \approx (N+13)/3$.

As remarked in [10] the minimum sampling period T_{min} can be found by adapting the algorithm for the *minimal cost-to-time ratio cycle problem* presented in [12]. The total number of computations required by this program is $M_2 = N^3 \cdot \log_2(2N^3F) + N \cdot E$, where E is the total number of edges in the data-flow graph, and F is the maximum number of arithemetical nodes between a pair of time-delay nodes. In the Gray-Markel lattice filter $E = 6 \cdot N$ and $F = N+2$.

Table I shows the values of M_1 and M_2 for the Gray-Markel lattice digital filter. Notice that it is only when $N \geq 7$ that $M_2 < M_1$. This fact, along with the fact that most digital filter structures (unlike the Gray-Markel lattice) do not possess directed paths between all pairs of time-delay nodes, and hence typically possess a far smaller total number of loops than indicated by (5), accounts for the quite acceptable performance of our algorithm, when employed for "real problems," even though in principle, one could expect to sometimes encounter unreasonably long computation times.

VII. APPENDIX B

The function S is defined recursively, for all positive integers, by

$$S(1) = 1$$

$$S(N) = \frac{1}{2}\sum_{M=1}^{N-1} \binom{N}{M} S(M)S(N-M) \text{ for } N \geq 2. \tag{6}$$

We shall prove that $S(N) = (2N-3)!!$, for $N \geq 2$, where we define $i!! = i(i-2)(i-4)\ldots 5 \cdot 3 \cdot 1$ for i odd. The proof is by induction. That is, for all $J = 1, \ldots, N-1$, we assume that $S(J) = (2J-3)!!$ and we shall show that $S(N) = (2N-3) \cdot S(N-1)$.

Proof: Using the property of binomial coefficients

$$\binom{N}{M} = \binom{N-1}{M-1} + \binom{N-1}{M} \quad \text{for } 1 \leq M \leq N-1$$

we find from (6)

$$S(N) = \frac{1}{2}\sum_{M=1}^{N-1}\left[\binom{N-1}{M-1} + \binom{N-1}{M}\right]S(M)S(N-M).$$

Define $M' = N-M$, then

$$S(N) = \frac{1}{2}\Big[\sum_{M'=1}^{N-1} \binom{N-1}{N-1-M'} S(N-M')S(M') + \sum_{M=1}^{N-1} \binom{N-1}{M} S(M)S(N-M)\Big].$$

Using the property of binomial coefficients

$$\binom{N-1}{N-1-M'} = \binom{N-1}{M'} \quad \text{for } 0 \leq M' \leq N-1 \tag{7}$$

we find

$$S(N) = \sum_{M=1}^{N-1} \binom{N-1}{M} S(M)S(N-M). \tag{8}$$

Let

$$S(N-M) = (2N-2M-3) \cdot S(N-M-1) \quad \text{for } 1 \leq M \leq N-2.$$

Then (8) becomes

$$S(N) = \binom{N-1}{N-1} S(N-1)S(1) + \sum_{M=1}^{N-2} \binom{N-1}{M} (2N-2M-3)S(M)S(N-1-M).$$

Define $M' = N-1-M$ and split the summation into two identical parts, then

$$S(N) = S(N-1) + \frac{1}{2}\Big[\sum_{M=1}^{N-2} \binom{N-1}{M}(2N-2M-3) \times S(M)S(N-1-M) + \sum_{M'=1}^{N-2} \binom{N-1}{N-1-M'}(2M'-1)\, S(N-1-M')S(M')\Big].$$

TABLE II

Name	Input 1	Input 2	;	Comments
A-Name	Input 1	Input 2	;	Adder
M-name	Input	Coef.	;	Multiplier
T-name	Input		;	Time delay
Y	Output		;	Output node, and last line

TABLE III

Name	Input 1	Input 2	;	Comments
M1	T1	b1		
M2	T2	b2		
M3	A1	a0		
M4	T1	a1		
M5	T2	a2		
A1	M1	A2		
A2	M2	X	;	X = input node
A3	M3	A4		
A4	M4	M5		
T1	A1			
T2	T1			
Y	A3			

Using property (7) and taking the corresponding terms together we have

$$S(N) = S(N-1) + (2N-4) \cdot \frac{1}{2} \sum_{M=1}^{N-2} \binom{N-1}{M} S(M)S(N-1-M).$$

We recognize the summation as $S(N-1)$, so

$$S(N) = (2N-3) \cdot S(N-1)$$

which completes the proof.

VIII. Appendix C

The program's input-file entries are of the form shown in Table II.

The input file for the second-order direct form II filter example is shown in Table III.

References

[1] A. Y. Kwentus *et al.*, "A programmable digital filter IC employing multiple processors on a single chip," *IEEE Trans. Circ. Syst. for Video Technology*, vol. 2, pp. 231-244, June 1992.

[2] A. Y. Kwentus *et al.*, "A ring-structured topology of programmable digital filter processors on a single chip," in *Proc. Int. Symp. on Sig., Syst. and Electronics*, pp. 278-281, Sept. 1992, also in *Proc. 1992 IEEE Workshop on VLSI Sig. Proc.*, pp. 195–204, Oct. 1992.

[3] D. A. Schwartz and T. P. Barnwell III, "Cyclo-static multiprocessor scheduling for the optimal realization of shift-invariant flow graphs," in *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Proc.*, pp. 1384–1387, 1985.

[4] D. A. Schwartz, "Synchronous multiprocessor realizations of shift invariant flow graphs," Ph.D. dissertation, Georgia Institute of Technology, 1985.

[5] D. A. Schwartz *et al.*, "The optimal synchronous cyclo-static array: a multiprocessor supercomputer for digital signal processing," in *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Proc.*, pp. 2891–2894, 1986.

[6] A. Fettweis, "Digital filters related to classical filter networks," *Arch. Elek. Übertragung.*, vol. 25, pp. 79–89, Feb. 1971.

[7] A. Fettweis, "Realizability of digital filter networks," *Arch. Elek. Übertragung.*, vol. 30, pp. 90–96, Feb. 1976.

[8] M. Renfors and Y. Neuvo, "The maximum sampling rate of digital filters under hardware speed constraints," *IEEE Trans. Circ. Syst.*, vol. CAS-28, pp. 196–202, Mar. 1981.

[9] M. Renfors and Y. Neuvo, "Fast multiprocessor realizations of digital filters," in *Proc. IEEE Int. Conf. Acoustics, Speech, Sig. Proc.*, pp. 916–919, 1980.

[10] S. H. Gerez *et al.*, "A polynomial-time algorithm for the computation of the iteration-period bound in recursive data-flow graphs," *IEEE Trans. Circ. Syst. I*, vol. 39, pp. 49–52, Jan. 1992.

[11] R. G. Busacker and T. L. Saaty, *Finite Graphs and Networks*. New York: McGraw-Hill, 1965.

[12] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart and Winston, 1976.

[13] C. E. Leiserson *et al.*, "Optimizing synchronous circuitry by retiming," in *Proc. Third Caltech Conf. VLSI*, Pasadena, CA, pp. 87–116, 1983.

[14] K. K. Parhi and D. G. Messerschmitt, "Static rate-optimal scheduling of iterative data-flow programs via optimum unfolding," *IEEE Trans Comput.*, vol. 40, pp. 178–195, Feb. 1991.

[15] S. M. Heemstra de Groot *et al.*, "Range-chart-guided iterative data-flow graph scheduling," *IEEE Trans. Circ. Syst. I*, vol. 39, pp. 351–364, May 1992.

[16] S. M. Heemstra de Groot and O. E. Herrmann, "Maximum throughput scheduling with limited resources for iterative data-flow graphs by means of the scheduling-range chart," in *Proc. Euromicro Workshop on Real Time Systems*, pp. 8–16, 1990.

[17] S. M. Heemstra de Groot and O. E. Herrmann, "Rate optimal scheduling of recursive DSP algorithms based on the scheduling range chart," in *Proc. IEEE Int. Symp. Circ. and Syst.*, pp. 1805–1808, 1990.

[18] S. M. Heemstra de Groot and O. E. Herrmann, "Evaluation of some multiprocessor scheduling techniques of atomic operations for recursive DSP graphs," in *Proc. Europ. Conf. Circ. Theory and Des.*, pp. 400-404, 1989.

[19] A. H. Gray, Jr. and J. D. Markel, "Digital lattice and ladder filter synthesis," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 491–500, Dec. 1973.

[20] J. Blazewicz, "Selected topics in scheduling theory," in *Surveys in Combinatorial Optimization*, P.L. Hammer, Ed., Amsterdam: North-Holland, pp. 1–59, 1987.

[21] W. A. Kohler, "A preliminary evaluation of the critical path method for scheduling tasks on multiprocessor systems," *IEEE Trans. Comput.*, vol. C-24, pp. 1235–1238, Dec. 1975.

[22] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Trans. Computers*, vol. C-36, pp. 24–35, Jan. 1987.

[23] E. A. Lee and D. G. Messerschmitt, "Synchronous data flow," *Proc. IEEE*, vol. 75, pp. 1235–1245, Sept. 1987.

[24] H. B. Voelcker and E. E. Hartquist, "Digital filtering via block recursion," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 169–176, June 1970.

Michael J. Werter (M '92) received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the Eindhoven University of Technology, the Netherlands, in 1982, 1984, and 1989, respectively.

From 1984 to 1989, he was a Scientific Research Fellow at the Eindhoven University of Technology. Since 1989, he has been employed at the University of California, Los Angeles, where he teaches courses and performs research in the areas of electric circuit analysis, electronics, and digital signal processing.

Dr. Werter's fields of interest include nonlinear effects in digital signal processing, multidimensional filtering, multiprocessor scheduling, and adaptive signal processing applications.

Alan N. Willson, Jr. (S '66–M'67–SM '73–F '78) received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, GA, in 1961, and the M.S. and Ph.D. degrees from Syracuse University, Syracuse, NY, in 1965 and 1967, respectively.

From 1961 to 1964 he was with IBM, Poughkeepsie, NY. He was an instructor of Electrical Engineering at Syracuse University from 1965 to 1967. From 1967 to 1973, he was a Member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. Since 1973, he has been with the faculty of the University of California, Los Angeles, where he is now Professor of Engineering and Applied Science in the Electrical Engineering Department. In addition, he served the UCLA School of Engineering and Applied Science as Assistant Dean for Graduate Studies from 1977 through 1981, and is currently Associate Dean of Engineering. He has been engaged in research concerning computer-aided circuit analysis and design, the stability of distributed circuits, properties of nonlinear networks, theory of active circuits, digital signal processing, analog circuit fault diagnosis, and integrated circuits for signal processing. He is Editor of the book *Nonlinear Networks: Theory and Analysis* (IEEE Press, 1974.)

Dr. Willson is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, the Society for Industrial and Applied Mathematics, and the American Society for Engineering Education. From 1977 to 1979, he served as Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. In 1980, he was General Chairman of the Fourteenth Asilomar Conference on Circuits, Systems, and Computers. During 1984, he served as President of the IEEE Circuits and Systems Society. He was a recipient of the 1978 Guillemin–Cauer Award of the IEEE Circuits and Systems Society, the 1982 George Westinghouse Award of the American Society for Engineering Education, the 1982 Distinguished Faculty Award of the UCLA Engineering Alumni Association, the 1984 Myril B. Reed Best Paper Award of the Midwest Symposium on Circuits and Systems, and the 1985 W. R. G. Baker Award of the IEEE.

AN IMPROVEMENT TO THE POWELL AND CHAU LINEAR PHASE IIR FILTERS *

A. N. Willson, Jr. and H. J. Orchard
University of California
Los Angeles, CA 90024-1594

Abstract

A technique using Jacobian elliptic functions is given which, by removing a previous method's double-zero constraint, yields improved designs of linear phase IIR filters.

Introduction

In theory it is possible to implement linear phase IIR filters as a tandem connection of an arbitrary transfer function $H(z)$ and a time-reversed version of the same function $H(z^{-1})$. Powell and Chau have devised a clever technique for doing this by approximating a local (in time) time-reversal operation using two copies of a desired IIR filter, several blocks of storage registers, and control circuitry that accesses two of these register blocks on a "last-in, first-out" (LIFO) basis. There is, however, a certain inefficiency in the use of identical transfer functions in the $H(z^{-1})H(z)$ cascade because the stopband transmission zeros all appear as "double zeros" in any such system. We have devised a design technique, using Jacobian elliptic functions, for an optimal pair of transfer functions which, when cascaded as $H_1(z^{-1})H_2(z)$ in the same type of IIR time-reversal system, will meet the same linear-phase design specifications as that of [1], but which also yields an additional 6 dB of stopband loss. This extra loss can be traded for lower passband ripple and/or a narrower transition band by a simple revision of the design specifications.

The technique of [1] of course yields only an *approximate* linear-phase design because it happens that certain errors are inevitable in the implementation of the time-reversal process due to the finite length of the register blocks and the infinite length of the filter's impulse response. For large enough register blocks the approximation can be quite acceptable, but this required length grows as the filter specifications become more demanding. When the register-block length is not quite sufficient, errors in the passband magnitude and phase response occur. Our investigations indicate that these effects are less pronounced in our $H_1(z^{-1})H_2(z)$ design than they are in the design of [1]; thus, this represents another advantage for our modified approach.

To illustrate our design technique consider the Fig. 1(a) pole-zero plot of a lowpass filter $H(z)$. The corresponding pole-zero pattern of $H(z^{-1})$ is shown in Fig. 1(b), and the overall filter designed as in [1] would have the pole-zero pattern shown in Fig. 1(c). In principle, a transfer function having the same set of Fig. 1(c) poles, but whose zeros (while confined to the unit circle) were not constrained to have order two, would be better able to distribute the zeros throughout the filter's stopband, thereby yielding a more efficient transfer function. Such a filter, whose pole-zero pattern could take the form of Fig. 1(d), would still possess a *linear phase* frequency response.

To improve the system given in [1] we must find a way to modify the elliptic filter design method to meet given passband and stopband specifications, such that it uses $2n$ simple zeros, distributed throughout the stopband, and uses n (i.e., half as many) *double* poles (thus, a total of $2n$ poles) inside the unit circle. Then, this collection of poles and zeros can be allocated to two transfer functions $H_1(z)$, $H_2(z)$ both having identical sets of (simple) poles. If we then build a filter with a transfer function $H(z) = H_1(z^{-1})H_2(z)$ it will have the pole-zero pattern of Fig. 1(d). Its implementation could use the same structure given in [1]; however, while an $H(z)H(z^{-1})$ system has linear phase for *any* $H(z)$, its Fig. 2 generalization has linear phase for all $H_1(z)$, $H_2(z)$ where (1) H_1 and H_2 have identical poles, and (2) the zeros of H_1 and H_2 all lie on the unit circle.

We have, in fact, found a method to design optimal transfer functions $H_1(z)$, $H_2(z)$ yielding a Fig. 2 filter with an equal-ripple passband and stopband. The functions $H(z) = H_1(z^{-1})H_2(z)$ are first constructed as their equivalents $\hat{H}(s) = \hat{H}_1(-s)\hat{H}_2(s)$ in the conventional analog variable $s = \Sigma + j\Omega$, where $z = (1+sT)/(1-sT)$. Reversing the sign of s corresponds to taking the reciprocal of z. The equal-ripple passband is normalized to $0 \le \Omega \le 1$, and the lower edge of the equal-ripple stopband is then at Ω_s.

We require $\hat{H}_1(s)$ and $\hat{H}_2(s)$ to have *identical* left-half s-plane poles, so the poles of $\hat{H}_1(-s)$ must lie in the right-half s-plane at the negatives of the poles of $\hat{H}_2(s)$. The poles of $\hat{H}(s)$ are thus the zeros of an even

*This work was supported by the National Science Foundation under Grant MIP-9201104 and by the Office of Naval Research under Grant N00014-91-J-1852 and by a grant from the State of California and TRW, through the California MICRO program.

0-7803-1775-0/94 $3.00 © 1994 IEEE

polynomial. The zeros of $\hat{H}(s)$ must be simple and lie on the $j\Omega$ axis in the stopband $\Omega_s \le \Omega \le \infty$. It follows that $\hat{H}(j\Omega)$ is purely real and of even degree.

If $\hat{H}_1(s)$ and $\hat{H}_2(s)$ had identical zeros as well as identical poles, then we should have $\hat{H}_1(s) \equiv \hat{H}_2(s)$, and $\hat{H}(s)$ would reduce to the conventional even function $\hat{H}_1(-s)\hat{H}_1(s)$ which is used in constructing a complex filter function $\hat{H}_1(s)$ to have a prescribed loss while ignoring the phase. We exploit this observation by using very much the same tools for constructing $\hat{H}(s)$ as one would for $\hat{H}_1(-s)\hat{H}_1(s)$ except for making it have simple rather than double $j\Omega$-axis zeros.

A Modified Elliptic Filter Design

We consider first the simplest case where all the zeros of $\hat{H}(s)$ are at infinity, and where we do have $\hat{H}_1(s) \equiv \hat{H}_2(s)$. Then $\hat{H}_1(s)$ is identical to the standard Chebyshev lowpass filter. This $\hat{H}(s)$, of degree $n = 2m$, can be constructed parametrically in the classic way with circular functions as:

$$\frac{1}{\hat{H}(s)} = 1 + \epsilon^2 \cos^2 m\theta \quad \text{and} \quad \Omega = \cos\theta \qquad (1)$$

The passband ripple of $\hat{H}(s)$ (not of $\hat{H}_1(s)$) is $\alpha_p = 20\log_{10}(1+\epsilon^2)$ dB

However, if we try to generalize (1) as it stands to the elliptic-function case, we would still retain $\hat{H}_1(s) \equiv \hat{H}_2(s)$ and $\hat{H}(s)$ would have double $j\Omega$-axis zeros. To circumvent this we rearrange (1), replacing $\cos^2 m\theta$ by $(\cos 2m\theta + 1)/2$ and $2m$ by n to get

$$\frac{1}{\hat{H}(s)} = 1 + \epsilon^2(\cos n\theta + 1)/2 = \frac{1}{1-t}(1 + t\cos n\theta)$$

where $1+\epsilon^2 = (1+t)/(1-t)$. For digital filters we can discard the factor $(1-t)^{-1}$ yielding

$$\frac{1}{\hat{H}(s)} = 1 + t\cos n\theta \quad \text{and} \quad \Omega = \cos\theta \qquad (2)$$

with passband ripple $\alpha_p = 20\log_{10}(1+t)/(1-t)$ dB.

It is easily confirmed that the poles of $\hat{H}(s)$ lie on an ellipse in the s-plane at the points

$$\begin{aligned} s_\sigma &= \gamma \sin\frac{(2\sigma-1)\pi}{n} + j\delta\cos\frac{(2\sigma-1)\pi}{n} \\ (\sigma &= 1,\, 2,\, \ldots,\, n) \end{aligned} \qquad (3)$$

where

$$\gamma = \sinh\theta_2,\ \delta = \cosh\theta_2,\ \exp n\theta_2 = t^{-1} + (t^{-2}-1)^{\frac{1}{2}}.$$

The poles of $\hat{H}_1(s) \equiv \hat{H}_2(s)$ are the left-half s-plane poles of $\hat{H}(s)$; their zeros are all at infinity.

To distribute simple zeros over the stopband, instead of having them all at infinity, we merely replace the cosine function in (2) by its appropriate Jacobian elliptic function equivalent, namely the $\mathrm{cd}(u;k)$ function (*not* the $\mathrm{cn}(u;k)$ function). Eq. (2) then becomes:

$$\frac{1}{\hat{H}(s)} = 1 + t\,\mathrm{cd}[nuK_1/K : k_1] \quad \text{and} \quad \Omega = \mathrm{cd}\,(u\,;\,k) \qquad (4)$$

where $k = \Omega_s^{-1}$. The Jacobian functions are doubly periodic. When, as in the present application, the modulus k is real and $0 < k < 1$, one period is real with quarterperiod K and one is imaginary with quarterperiod K'. The quarterperiods for elliptic functions are the counterparts of $\pi/2$ for circular and hyperbolic functions. Varying k changes both K *and* the ratio K'/K.

In order to make the parametric equations in (4) describe a *rational* function with the required characteristics, it is necessary that the modulus k_1 (whose associated quarterperiods are K_1 and K_1') and the scale factor nK_1/K on u be chosen so that, from the viewpoint of the variable u, the quarterperiod rectangle of $1/\hat{H}(s)$ fits exactly n times along the real axis into the quarterperiod rectangle of the Ω, but only once along the imaginary axis. That is, so that when $u = K$, nuK_1/K becomes nK_1, while when $u = jK'$, nuK_1/K becomes jK_1'. This requires the modulus k_1 to be related to k so that $nK'/K = K_1'/K_1$. This can be expressed alternatively by $q^n = q_1$ in terms of the parameter q belonging to the closely related Theta functions and defined by $q = \exp(-\pi K'/K)$.

Over the stopband, $\Omega_s \le \Omega \le \infty$, $1/\hat{H}(s)$ has simple poles at which it changes sign. At Ω_s and at the turning points between the poles, the elliptic function has the value $\pm k_1^{-1}$. In an interval between two adjacent poles where the function is positive, the minimum loss is $20\log_{10}(1+t/k_1)$ dB, whereas when it is negative the minimum loss is $20\log_{10}(t/k_1 - 1)$ dB. In any practical filter t/k_1 will be much larger than unity, so the minimum loss can be approximated by:

$$\alpha_s = 20\log_{10}\frac{t}{k_1}\ \text{dB} \qquad (5)$$

The exact minima of the loss will be alternately slightly higher and slightly lower than (5), but the departures from (5) are quite small. For example, at 40 dB loss they are less than 0.1 dB, and at 60 dB less than 0.01 dB.

At the outset in a design, α_p, α_s and k are prescribed and we need to find the lowest degree n of filter that meets this specification. As n must be an integer, even its lowest permissible value will usually provide some margin in performance, and the parameters can then be readjusted to distribute this margin over α_p, α_s and k. Eq. (5). combined with some way of computing k_1 from k and n, gives the relationship between the four quantities concerned.

By expanding k_1^2 into a power series in $q_1 = \exp(-\pi K_1'/K_1)$, and then replacing q_1 by q^n, we get $k_1^2 = 16q^n - 128q^{2n} + 704q^{3n} - \ldots$. When $\alpha_p \le 0.1$ dB and $\alpha_s \ge 20$ dB, $q_1 = q^n \le 2.1 \times 10^{-8}$ and it is clear

that the first term in the series is a more than adequate practical approximation to k_1^2. Substituting $4q^{n/2}$ for k_1 in (5) leads to the design equation

$$\alpha_s = n\ 10\log_{10}\frac{1}{q} + 20\log_{10}t - 12.04\ \text{dB} \qquad (6)$$

The parameter q depends only upon k and is computed as follows:
Let $k = \sin\phi$.

$$\text{If } \phi \leq 45^\circ \qquad q = \frac{1}{2}\ \frac{1-\sqrt{\cos\phi}}{1+\sqrt{\cos\phi}}$$

$$\text{If } \phi > 45^\circ \qquad q' = \frac{1}{2}\ \frac{1-\sqrt{\sin\phi}}{1+\sqrt{\sin\phi}}$$

and

$$q = \exp\left[\frac{\pi^2}{\ln q'}\right]$$

This gives q accurate to at least 1 part in 10^5.

When k, α_p and n have been chosen, there remains only the calculation of the complex poles and $j\Omega$-axis zeros of $\hat{H}(s)$ to complete the design. The simplest and most accurate way of doing this is via a sequence of Landen transformations from the corresponding circular functions. The Landen transformation is an algebraic relation between certain elliptic functions belonging to two different modulus values with the ratio K'/K for one modulus twice that for the other. Iterating the transformation soon produces a modulus so small, and for which the ratio K'/K is so large, that the elliptic functions belonging to it have degenerated into (are numerically indistinguishable from) circular functions. Working backwards along this chain of descending moduli one can then, step by step, transform the circular functions into the desired elliptic functions.

Let us denote the initial modulus $k = \Omega_s^{-1}$ by k_0 and the moduli obtained by successive Landen transformations by k_1, k_2, ... The k_i are related by $k_{i+1} = \left[k_i/\left(1+\sqrt{1-k_i^2}\right)\right]^2$. If the arithmetic is carried out to d decimal digits, then the transformations are stopped when $k_r \leq 10^{-d}$.

Next, we find the reciprocals of the complex poles s_σ given in (3) for the circular-function case to which the elliptic functions have been reduced after r steps of the Landen transformation. Let $a_r + jb_r = s_\sigma^{-1}$. After r transformations using

$$a_{i-1} + jb_{i-1} = \frac{1}{1+k_i}\left[a_i + jb_i - \frac{k_i}{a_i + jb_i}\right] \qquad (7)$$

we get $a_0 + jb_0$ which is the reciprocal of the corresponding pole of $\hat{H}(s)$. This need be done of course only for the left-half s-plane poles of $\hat{H}(s)$.

Finally, the $j\Omega$-axis zeros of $\hat{H}(s)$ are given by

$$\pm j\Omega_\sigma = \frac{\pm j}{k\ \text{cd}\left[(2\sigma-1)K/n;\ k\right]} \quad (\sigma = 1,\ 2,\ \ldots,\ n/2)$$

We use the same chain of moduli k_i as for the poles, and (7) with $a_r = 0$ and $b_r = 1/\cos\left[(2\sigma-1)\pi/(2n)\right]$. Starting with $a_r = 0$ causes all a_i to vanish, and (7) simplifies to

$$b_{i-1} = \frac{1}{1+k_i}\left[b_i + \frac{k_i}{b_i}\right] \qquad (8)$$

After r steps using (8), $\Omega_\sigma = b_0/k$.

The zeros of $\hat{H}(s)$ have to be split into two groups, one belonging to $\hat{H}_1(s)$ and one to $\hat{H}_2(s)$. The simplest approach is to arrange that the zeros of $\hat{H}_1(s)$ and $\hat{H}_2(s)$ interlace, but slight digressions from this may in some cases prove advantageous. We note that the separate functions $\hat{H}_1(s)$ and $\hat{H}_2(s)$ will not have flat passbands, although their tandem connection will. When $n/2$, the common degree of $\hat{H}_1$ and $\hat{H}_2$, is odd, both functions must have a zero at infinity. This requires $\hat{H}$ to have two zeros at infinity. But $\hat{H}$ defined by (4) has $n/2$ conjugate $j\Omega$-axis pairs of zeros and none at infinity. As $\hat{H}$ is an even function of s we correct this by making a bilinear transformation on Ω^2 $(= -s^2)$ moving to infinity the conjugate pair of zeros nearest infinity, while keeping the passband edge frequencies $\Omega^2 = 0$ and $\Omega^2 = 1$ fixed. The transformation is $\tilde{\Omega}^2 = \Omega^2\left(\Omega_m^2 - 1\right)/\left(\Omega_m^2 - \Omega^2\right)$ where $\tilde{\Omega}$ is the transformed frequency and Ω_m is the largest zero of $\hat{H}$ as given by (4). This will increase slightly the stopband edge from k^{-1} to $[k\ \text{cd}\,(K/n\ ;\ k)]^{-1}$. We compensate by chosing k in (4) so that $k\ \text{cd}\,(K/n\ ;\ k) = \Omega_s^{-1}$ instead of $k = \Omega_s^{-1}$. The value of k can easily be found via the iteration $k_{r+1} = k_0/\text{cd}\,(K/n\ ;\ k_r)$, starting with $k_0 = \Omega_s^{-1}$. This transformation will reduce slightly the 6.02 dB increase of loss we would otherwise get; the reduction varies from 1.075 dB when $n = 10$ to 0.357 dB when $n = 30$. $H_1(z)$ and $H_2(z)$ can be found by using $z = (1+sT)/(1-sT)$.

Example

Fig. 3 shows an example from [1]. Here the result corresponding to $|H_1(z^{-1})H_2(z)|$, for $z = e^{j\omega}$, using our modified design, is plotted as a dashed curve, while the corresponding plot for $|H(z)|^2$ from [1] (example 3 in Table I) is the solid curve. As expected, our design yields an improved stopband. We have proved that we *always* obtain a stopband having approximately 6 dB ($\approx$ 0.7 nepers) more attenuation than we obtain by the technique of [1] employing $|H(z)|^2$.

In Fig. 4 we examine the passband errors for the Fig. 3 example. Notice that compatible scales have been used in both Fig. 4(a) and Fig. 4(b): that is, we measure the gain in nepers and the phase in radians. (Notice also the 10^{-6} factor on both vertical axes in Fig. 4.) These results, which are typical of those for all examples considered, indicate that the non-ideal passband errors are less pronounced in filters resulting from our $H_1(z^{-1})H_2(z)$ design technique than they are for those obtained through the design technique of [1].

Concluding Remark

The increase in stopband loss of approximately 6 dB, caused by separating the double zeros, may be reminiscent of a similar feature in the relationship between certain minimum-phase and linear-phase FIR filters, as described in [2]. From our perspective here, however, this similarity is superficial; the FIR filter design techniques of [2] unfortunately provide no help in solving our IIR filter design problem.

Acknowledgment

We would like to acknowledge the work of Dr. M. Werter, who wrote the computer program that simulates the filters described in this paper.

References

[1] S. R. Powell and P. M. Chau, "A technique for realizing linear phase IIR filters," *IEEE Transactions on Signal Processing*, vol. 39, pp. 2425-2435, Nov. 1991.

[2] O. Herrmann and H. W. Schüssler, "Design of nonrecursive digital filters with minimum phase," *Electron. Lett.*, vol. 6, pp. 329-330, 1970.

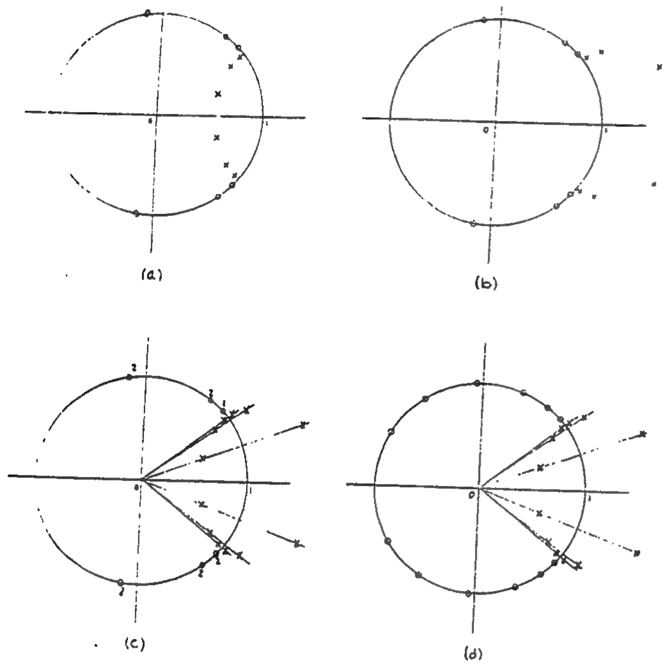


Fig. 1. Pole-zero patterns illustrating modified Linear-Phase IIR filter design technique. (a) Poles and zeros for $H(z)$. (b) Poles and zeros for $H(z^{-1})$; (c) Poles and zeros of $H(z^{-1})H(z)$; (d) Poles and zeros of $H_1(z^{-1})H_2(z)$.

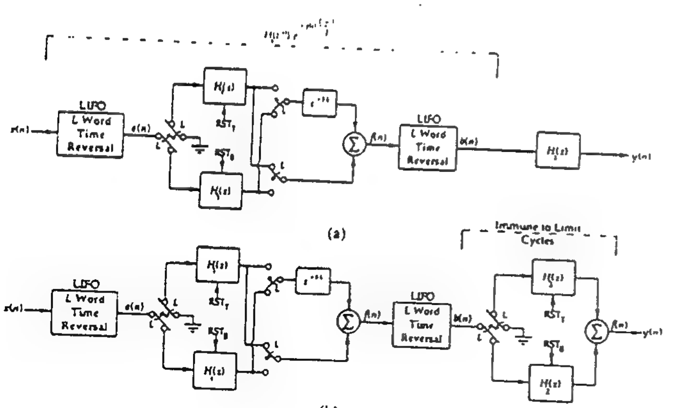


Fig. 2. Linear-phase filter implementing $H_1(z^{-1})H_2(z)e^{-j\omega(L-1)/2}$. (a) Simple implementation. (b) Limit-cycle free implementation.

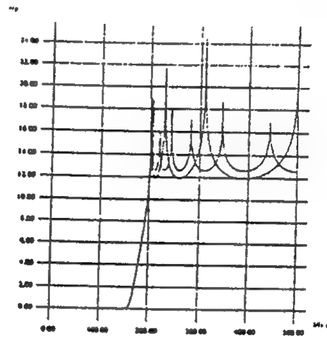


Fig. 3. Comparison of linear-phase filter design methods. Solid line: technique of Ref. [1]. Dashed line: presently proposed modification.

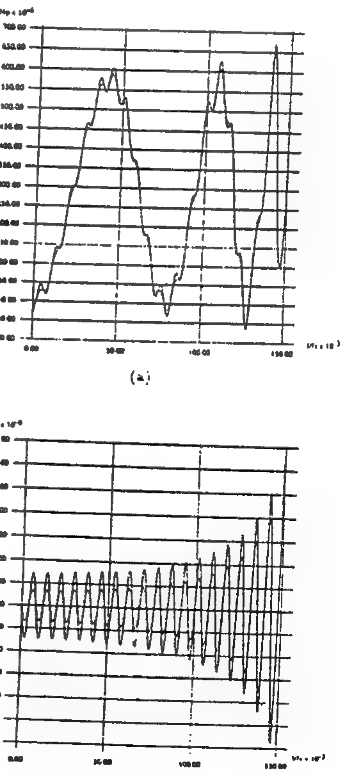


Fig. 4. Comparison of passband gain and phase response errors for the example filters of Fig. 3: (a) gain response errors (in nepers); (b) phase response deviation from exactly linear phase (in radians). For both cases, solid line: technique of Ref. [1]; Dashed line: presently proposed modification.

The Design of Two-Channel Lattice-Structure Perfect-Reconstruction Filter Banks Using Powers-of-Two Coefficients

Bor-Rong Horng, Henry Samueli, and Alan N. Willson, Jr.

***Abstract*—An optimization technique is presented for the design of two-channel lattice-structure perfect-reconstruction filter banks with powers-of-two coefficients. The filter coefficients are represented by a canonic signed-digit (CSD) code. The proposed technique requires the original optimal infinite-precision coefficients as the starting point, and searches for the set of CSD coefficients that minimizes the peak stopband ripple. Design examples are given to show that perfect-reconstruction filter banks with good filtering performance can be obtained.**

I. Introduction

Multirate analysis/synthesis filter banks find application in many areas [1]. Recently, much attention has been given to the design of multiplierless filter banks with applications to subband coding [2]–[4]. In [3], three sets of short-tap filter banks with powers-of-two coefficients were derived by judiciously factoring a seven-tap half-band product filter. Although the perfect-reconstruction property is preserved for these filter banks, the coding gain is poor due to poor filtering performance [5]. In [4], a canonic-signed digit (CSD) code search technique was used to design multiplierless filter banks with good filtering performance at the expense of the perfect-reconstruction property. Although this technique can achieve negligible signal-reconstruction error in practical applications [4], the design of perfect-reconstruction filter banks with good filtering performance using powers-of-two coefficients is yet open and of great interest.

Recently, several novel lattice-structure perfect-reconstruction filter banks have been reported [6]–[9]. One desirable feature of these lattice-structure filter banks is that the perfect-reconstruction property is preserved, even under the quantization of the lattice coefficients. This feature opens the door to the design of multiplierless perfect-reconstruction filter banks with good filtering performance since we need only to find the set of CSD lattice coefficients yielding the desired filtering performance. However, not all of these filter banks are well suited for CSD design. As reported in [8], the dynamic range of the optimal coefficients is too wide for lattice-structure perfect-reconstruction filter banks employing linear-phase filters. A prohibitive number of digits would be required to implement such filter banks using fixed-point arithmetic with current technologies. Therefore, these filters are not suitable for CSD design. The filter banks in [6], however, do have a good, small coefficient dynamic range, and should be good candidates for CSD design. In this paper, we examine the use of an optimization technique, which adopts a two-stage local search strategy over the CSD code [10], to optimize the performance of such filter banks. Such designs should lead to computationally efficient, multiplierless perfect-reconstruction filter banks which should be more useful, in practice, than the original infinite-precision design [6].

Manuscript received December 18, 1991. This work was supported in part by the Office of Naval Research under Grant N00014-91-J-1852, and in part by the National Science Foundation under Grant MIP-9201104. This paper was recommended by Associate Editor M. A. Soderstrand.

B.-R. Horng was with the Department of Electrical Engineering, University of California, Los Angeles, CA. He is now with the Digital Communications Division, Rockwell International, Newport Beach, CA 92658.

H. Samueli and A. N. Willson, Jr. are with the Integrated Circuits and Systems Laboratory, Department of Electrical Engineering, University of California, Los Angeles, CA 90024.

IEEE Log Number 9210024.

II. Optimization Algorithm

Before formulating the optimization procedure, let us make some observations.

1) Although the original infinite-precision lattice filter banks have monotonically decreasing stopband peak error [6], our computer simulations show that after rounding the lattice coefficients to the nearest CSD code, the peak error in the stopband is no longer monotone decreasing. Therefore, the original criterion of minimizing the stopband integrated squared error would be inappropriate here. Furthermore, the original lattice filter banks have low passband sensitivity. Thus, a reasonable objective function to be minimized would simply be the peak error in the stopband of the lowpass filter:

$$\delta = \max_{\omega_s \le \omega \le \pi} |H_0(e^{j\omega})|. \tag{1}$$

2) The impulse response coefficients of $H_0(z)$ and $H_1(z)$ are products of the lattice coefficients; thus, the shape of the frequency response would be affected if we scale the lattice coefficients. In other words, large scaling on the lattice coefficients would not improve the frequency response. Our computer simulations show that only a little fine scaling of the original optimal point might help. Furthermore, the quantization error of each lattice coefficient would accumulate on its corresponding impulse response coefficient. Therefore, we would like to make the quantization error of each lattice coefficient as small as possible. One possibility is to search for only the fractional part of each lattice coefficient which would limit the quantization error within the fractional part of each lattice coefficient. This, of course, fails to decrease the number of nonzero digits for those lattice coefficients with integer part. However, as can be observed in [6], such coefficients tend to be few in the original optimal infinite-precision coefficient design, and thus the hardware penalty is minor.

Based on the above observations, we propose the following optimization procedure.

1) The optimal infinite-precision lattice coefficients in [6] are used as the starting point.
2) The computer program of [10] is modified to search for the set of CSD lattice coefficients such that (1) is minimized. Notice that we only search for the fractional part of each lattice coefficient, and only fine scaling on the original optimal lattice coefficients is performed.

III. Design Examples

Example 1: The filter bank denoted 32E in [6] was designed. The low-pass magnitude response plots of the original 32E filter and the CSD design are shown in Fig. 1. As reported in [6], the original 32E has monotonic decreasing stopband ripples, with the first-peak stopband attenuation of 25 dB, whereas the CSD design has a quasi-equal-ripple stopband with a minimum stopband attenuation of 29 dB. Two nonzero digits were chosen for the optimization. The CSD code of the lattice coefficients is shown in Table I. It is interesting to observe that only a single adder/subtractor, on average, is required to implement each lattice coefficient for this example.

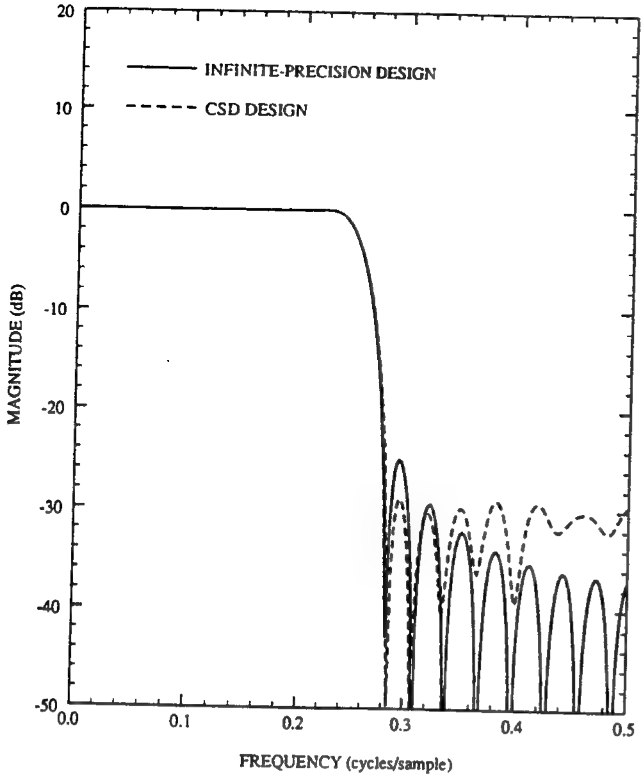


Fig. 1. The low-pass magnitude response plots of the original lattice-structure filter bank 32E and CSD design in Example 1.

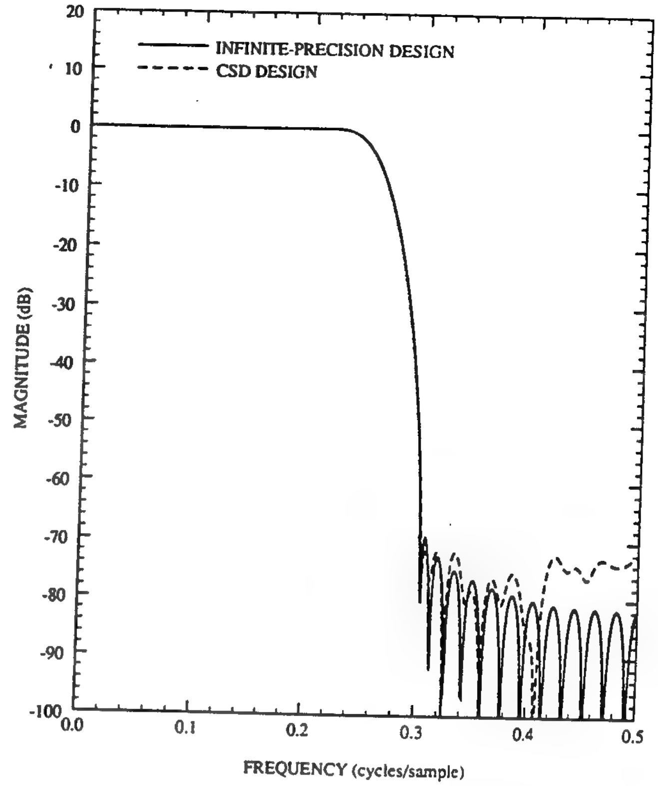


Fig. 2. The low-pass magnitude response plots of the original lattice-structure filter bank 48F and the CSD design in Example 2.

TABLE I
CSD CODE OF α_m IN EXAMPLE 1

m	α_m
1	$-2^{1} - 2^{-0} + 2^{-5}$
2	$2^{0} - 2^{-4}$
3	$-2^{-1} + 2^{-5}$
4	$2^{-1} - 2^{-3}$
5	$-2^{-2} - 2^{-6}$
6	$2^{-2} - 2^{-5}$
7	$-2^{-3} - 2^{-5}$
8	$2^{-3} + 2^{-7}$
9	$-2^{-3} + 2^{-6}$
10	$2^{-4} + 2^{-6}$
11	$-2^{-4} - 2^{-8}$
12	$2^{-4} - 2^{-7}$
13	$-2^{-5} - 2^{-7}$
14	$2^{-5} - 2^{-8}$
15	$-2^{-5} + 2^{-7}$
16	2^{-6}

TABLE II
CSD CODE OF α_m IN EXAMPLE 2

m	α_m
1	$-2^{2} - 2^{1} - 2^{-4} + 2^{-8} + 2^{-10} + 2^{-12}$
2	$2^{1} + 2^{-13}$
3	$-2^{0} - 2^{-2} + 2^{-4} + 2^{-7} + 2^{-9} - 2^{-11}$
4	$2^{0} - 2^{-2} + 2^{-4} + 2^{-7} - 2^{-10}$
5	$-2^{-1} - 2^{-3} + 2^{-7} + 2^{-9}$
6	$2^{-1} - 2^{-6} - 2^{-8} + 2^{-10} + 2^{-12}$
7	$-2^{-1} + 2^{-3} - 2^{-6} + 2^{-8} + 2^{-11}$
8	$2^{-2} + 2^{-4} + 2^{-9} - 2^{-11} - 2^{-14}$
9	$-2^{-2} - 2^{-7} + 2^{-10} + 2^{-12} + 2^{-14}$
10	$2^{-2} - 2^{-5} - 2^{-7} - 2^{-10} - 2^{-12}$
11	$-2^{-2} + 2^{-4} + 2^{-6} + 2^{-9} - 2^{-11}$
12	$2^{-3} + 2^{-6} - 2^{-8} + 2^{-11} + 2^{-13}$
13	$-2^{-3} + 2^{-6} + 2^{-11} - 2^{-14}$
14	$2^{-3} - 2^{-5} - 2^{-7} - 2^{-10} - 2^{-13}$
15	$-2^{-4} - 2^{-9} - 2^{-14}$
16	$2^{-4} - 2^{-6} + 2^{-10} - 2^{-12} - 2^{-14}$
17	$-2^{-5} - 2^{-9} - 2^{-11} - 2^{-13}$
18	$2^{-5} - 2^{-7} - 2^{-11} + 2^{-14}$
19	$-2^{-6} + 2^{-10} - 2^{-12} + 2^{-14}$
20	$2^{-7} + 2^{-10} + 2^{-12} - 2^{-14}$
21	$-2^{-8} - 2^{-10} - 2^{-13}$
22	$2^{-9} + 2^{-11}$
23	$-2^{-10} - 2^{-14}$
24	$2^{-12} + 2^{-14}$

Example 2: The filter bank denoted 48F in [6] was designed. The low-pass magnitude response plots of the original 48F filter and the CSD design are shown in Fig. 2. As reported in [6], the minimum stopband attenuation of the original 48F filter is 70 dB. For such a large attenuation, more coefficient precision is needed. Therefore, more nonzero CSD digits are required. Five nonzero digits were chosen for the optimization, and the CSD code for the lattice coefficients is shown in Table II. The CSD design has a minimum stopband attenuation of 71.6 dB. Compared to the optimal equiripple design by Smith and Barnwell [6], [11], where a minimum stopband attenuation of 72 dB was reported, the CSD design is only 0.4 dB less. The CSD design requires 3.17 adders/subtractors, on average, to implement each lattice coefficient.

IV. CONCLUSIONS

We have presented an optimization technique for the design of two-channel lattice-structure perfect-reconstruction filter banks with CSD coefficients. The two-stage local search strategy in [10] has been successfully modified to search for the set of CSD lattice coefficients which minimizes the stopband peak error of the filter banks. Design examples have been given to show the effectiveness of the proposed algorithm.

REFERENCES

[1] P. P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," *Proc. IEEE*, vol. 78, pp. 56–93, Jan. 1990.

[2] G. Pirani, F. Rusina, and V. Zingarelli, "Multiplication-free filters for subband coding of speech," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1982, pp. 848–851.

[3] D. Le Gall and A. Tabatabai, "Sub-band coding of digital images using symmetric short kernel filters and arithmetic coding techniques," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Apr. 1988, pp. 761–764.

[4] B. R. Horng and A. N. Willson, Jr., "The design of multiplierless two-channel linear-phase FIR filter banks with applications to image subband coding," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 1990, pp. 650–653.

[5] J. C. Darragh, "Subband and transform coding of images," Ph.D. dissertation, Dep. Elec. Eng., UCLA, 1989.

[6] P. P. Vaidyanathan and P.-Q. Hoang, "Lattice structure for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 81–94, Jan. 1988.

[7] T. Q. Nguyen and P. P. Vaidyanathan, "Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 676–690, May 1989.

[8] ——, "Structures for M-channel perfect-reconstruction FIR QMF banks which yield linear-phase analysis filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 433–446, Mar. 1990.

[9] M. Vetterli and D. Le Gall, "Perfect reconstruction FIR filter banks: Some properties and factorizations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1057–1071, July 1989.

[10] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

[11] M. J. T. Smith and T. P. Barnwell, "Exact reconstruction techniques for tree structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434–441, June 1986.

Lagrange Multiplier Approaches to the Design of Two-Channel Perfect-Reconstruction Linear-Phase FIR Filter Banks

Bor-Rong Horng, *Member, IEEE*, and Alan N. Willson, Jr., *Fellow, IEEE*

Abstract—Two new approaches are presented for the design of two-channel perfect-reconstruction FIR filter banks employing linear-phase filters. We first formulate the optimization of perfect-reconstruction filter banks as a quadratic programming problem with linear constraints, and then as one with nonlinear constraints. Closed-form solutions for the first approach, and for the iteration problem in the second approach are obtained. Design examples for both approaches are given.

I. Introduction

Multirate filter banks are used in applications such as speech coding, TDM-FDM transmultiplexing, and image coding [1], [2]. In these analysis/synthesis systems, perfect-reconstruction filter banks have been reported recently [3]–[6]. When applied to low-rate subband image coding, the symmetric extension method [7], [8] has been shown to outperform the circular convolution method [2], and to yield both objective and subjective quality improvement at image boundaries. The symmetric extension method requires linear-phase analysis/synthesis filters; therefore perfect-reconstruction filter banks with linear-phase filters are desired in subband image coding. The design of two-channel perfect-reconstruction filter banks employing linear-phase filters has been reported recently [4], [9], [10], [11]. As discussed in [9], the factorization method and the complementary filter method might yield filters with poor quality. Novel lattice structures are reported in both [10] and [11], and in [11] it is reported that good-quality filters have been obtained by optimizing the lattice parameters.

In this paper, we present two new approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks. Both approaches analyze and design on the impulse responses of the analysis filter bank directly. The synthesis filter bank is then obtained by simply changing the signs of odd-order coefficients in the analysis filter bank. Our first approach deals with unequal-length filter banks. By designing the lower length filters first we can take advantage of the fact that the number of variables for designing the higher length filters is more than the number of perfect-reconstruction constraint equations. We thus formulate the design problem as a quadratic programming problem with linear constraints, and we use the Lagrange multiplier method, as described in [12] and [13], to obtain the closed-form solution for designing the higher length filters. Our second approach generalizes the first, and covers the design for all pairs of linear-phase perfect-reconstruction analysis filters. It formulates the design problem as a quadratic programming problem with nonlinear constraints. The Lagrange–Newton method is used to obtain the closed-form solution for the linearized iteration problem in the second approach. Design examples for both approaches are given.

A generic two-channel FIR filter bank is shown in Fig. 1, where $H_0(z)$ and $H_1(z)$ represent the low-pass and high-pass filters in the analysis bank, respectively, and $G_0(z)$ and $G_1(z)$ are the synthesis filters. Assuming perfect channels and codecs, it is well known that we can relate the reconstructed signal $\hat{x}(n)$ to the input signal $x(n)$ by

$$\hat{X}(z) = \tfrac{1}{2}[H_0(z)G_0(z) + H_1(z)G_1(z)]X(z) + \tfrac{1}{2}[H_0(-z)G_0(z) + H_1(-z)G_1(z)]X(-z).$$

Furthermore, by choosing

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} \doteq \begin{bmatrix} 2H_1(-z) \\ -2H_0(-z) \end{bmatrix}$$

we have

$$\hat{X}(z) = [H_0(z)H_1(-z) - H_0(-z)H_1(z)]X(z).$$

If we impose the following pure-delay constraint

$$H_0(z)H_1(-z) - H_1(z)H_0(-z) = z^{-2k+1} \tag{1}$$

then

$$\hat{X}(z) = z^{-2k+1}X(z).$$

Thus, we obtain a perfect-reconstruction system where the output $\hat{x}(n)$ is a delayed replica of the input $x(n)$.

Manuscript received May 7, 1990; revised January 17, 1991. This work was supported by the National Science Foundation under Grant MIP 86-03639 and by the Office of Naval Research under Grant N00014-91-J-1852.

B.-R. Horng was with the Electrical Engineering Department, University of California, Los Angeles, CA 90024. He is now with Rockwell International Corporation, Newport Beach, CA 92658-8902.

A. N. Willson, Jr., is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90024-1594.

IEEE Log Number 9104888.

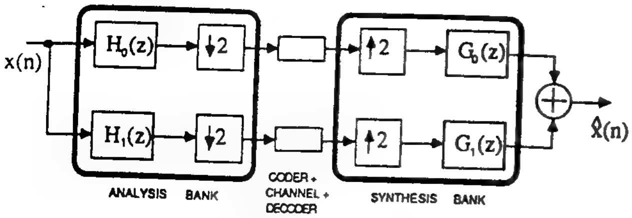


Fig. 1. Two-channel analysis and synthesis filter bank.

Combining the pure-delay constraint (1) and the linear-phase condition, it has been shown [4], [11] that only two types of systems yield nontrivial analysis filters:

1) both filters have even length and opposite symmetry, denoted type A systems in [11];

2) both filters have odd length and are symmetric, denoted type B systems in [11].

Using terminology defined in [14], for type A systems the analysis filters are either case 2 or case 4 since the lengths are even. It should also be noted that case 2 cannot realize a high-pass filter. Therefore, $H_1(z)$ must be case 4 and $H_0(z)$ must be case 2.

For type B systems it is obvious that both $H_0(z)$ and $H_1(z)$ must be case 1.

Furthermore, by examining the pure-delay constraint in (1), and the coefficient symmetry/antisymmetry of linear-phase filters, we can make the following observations (assuming the lengths of $h_0(n)$ and $h_1(n)$ to be N_0 and N_1, respectively):

1) the sum of the lengths must be a multiple of 4 [11];

2) the number of independent constraint equations in (1), k, is given by

$$k = \frac{N_0 + N_1}{4} \tag{2}$$

and $(N_0 + N_1)/2 - 1$ is the system delay;

3) the constraint equations can be expressed as

$$\frac{1}{2}\delta\left(i - \frac{N_0 + N_1}{4}\right) = \sum_{k=0}^{2i-1} (-1)^k h_0(2i - 1 - k) h_1(k)$$

$$i = 1, 2, \cdots, \frac{N_0 + N_1}{4}.$$

It should be noted here that for type B systems $h_0(n) = 0$ for $n \geq N_0$ and $h_1(n) = 0$ for $n \geq N_1$. By adding the coefficient symmetry/antisymmetry of the linear-phase filters the above equations can be expressed in the matrix form

$$\tilde{C}\tilde{y}_1 = m \tag{3}$$

where $\tilde{y}_1$ is an $(l_1 + 1)$-dimensional column vector

$$\tilde{y}_1 = [h_1(0) h_1(1) \cdots h_1(l_1)]^T$$

m is an $(N_0 + N_1)/4$-dimensional column vector

$$m = [0 \ \cdots \ 0 \ \tfrac{1}{2}]^T$$

$\tilde{C}$ is an $(N_0 + N_1)/4$-by-$(l_1 + 1)$ matrix with the elements formed by $h_0(n)$, $n = 0, 1, \cdots, l_0$ and

$$\begin{cases} l_0 = \dfrac{N_0}{2} - 1, \quad l_1 = \dfrac{N_1}{2} - 1; & \text{type A} \\ l_0 = \dfrac{N_0 - 1}{2}, \quad l_1 = \dfrac{N_1 - 1}{2}; & \text{type B.} \end{cases}$$

The next section addresses the formulation and design examples of our first approach, the Lagrange multiplier method. Section III addresses our second approach, the Lagrange–Newton method.

II. Lagrange Multiplier Method

This method deals with unequal-length filter banks; without loss of generality, we assume $N_0 < N_1$. The design process starts with the design of $H_0(z)$, which is case 2 for type A systems, or case 1 for type B systems, using any desired design criterion. Then, its coefficients are used as known variables in (1), yielding a set of linear constraint equations for designing $H_1(z)$. We recall that $H_1(z)$ is case 4 for type A systems, or case 1 for type B systems. Therefore, by defining

$$a_1(n) = 2h_1\left(\frac{N_1}{2} - n\right), \qquad n = 1, 2, \cdots, \frac{N_1}{2}$$

and

$$b_1(n) = \begin{cases} h_1\left(\dfrac{N_1 - 1}{2}\right), & n = 0 \\ 2h_1\left(\dfrac{N_1 - 1}{2} - n\right), & n \neq 0 \end{cases}$$

we can express the zero-phase frequency response of $H_1(z)$ as a scalar product [14]

$$H_1^*(e^{j\omega}) = y_1^T s_1(\omega)$$

where

$$y_1 = \begin{cases} \left[a_1(1) \ \ a_1(2) \cdots a_1\left(\dfrac{N_1}{2}\right)\right]^T; & \text{type A} \\ \left[b_1(0) \ \ b_1(1) \cdots b_1\left(\dfrac{N_1 - 1}{2}\right)\right]^T; & \text{type B} \end{cases}$$

and

$$s_1(\omega) = \begin{cases} \left[\sin\dfrac{\omega}{2} \ \ \sin\dfrac{3\omega}{2} \cdots \sin\dfrac{N_1 - 1}{2}\omega\right]^T; \\ \text{type A} \\ \left[1 \ \ \cos\omega \ \ \cos 2\omega \cdots \cos\dfrac{N_1 - 1}{2}\omega\right]^T; \\ \text{type B.} \end{cases}$$

The objective function to be minimized is

$$\Phi = \frac{1}{2\pi}\left\{\int_0^{\omega_{s1}} [H_1^*(e^{j\omega})]^2\, d\omega + \int_{\omega_{p1}}^{\pi} [1 - H_1^*(e^{j\omega})]^2\, d\omega\right\}$$

$$= \frac{1}{2} y_1^T Q y_1 + p^T y_1 + d$$

where

$$Q = \frac{1}{\pi}\left[\int_0^{\omega_{s1}} s_1(\omega)s_1^T(\omega)\, d\omega + \int_{\omega_{p1}}^{\pi} s_1(\omega)s_1^T(\omega)\, d\omega\right]$$

$$p^T = -\frac{1}{\pi}\left[\int_{\omega_{p1}}^{\pi} s_1^T(\omega)\, d\omega\right]$$

and

$$d = \frac{\pi - \omega_{p1}}{2\pi}.$$

For type A systems the elements of Q are given by

$$q(i, j) = \frac{1}{\pi}\left\{\int_0^{\omega_{s1}} \sin\left[\left(i - \frac{1}{2}\right)\omega\right]\sin\left[\left(j - \frac{1}{2}\right)\omega\right] d\omega + \int_{\omega_{p1}}^{\pi} \sin\left[\left(i - \frac{1}{2}\right)\omega\right]\sin\left[\left(j - \frac{1}{2}\right)\omega\right] d\omega\right\}$$

$$1 \le i, j \le \frac{N_1}{2}$$

$$= \begin{cases} \dfrac{\pi + \omega_{s1} - \omega_{p1}}{2\pi} + \dfrac{\sin[(2i - 1)\omega_{p1}] - \sin[(2i - 1)\omega_{s1}]}{(4i - 2)\pi}, & i = j \\[2ex] \dfrac{\sin[(i - j)\omega_{s1}] - \sin[(i - j)\omega_{p1}]}{2(i - j)\pi} - \dfrac{\sin[(i + j - 1)\omega_{s1}] - \sin[(i + j - 1)\omega_{p1}]}{2(i + j - 1)\pi}, & i \neq j \end{cases}$$

and the elements of $p^T = [p_1 \;\; p_2 \;\; \cdots \;\; p_{N_1/2}]$ are given by

$$p_i = -\frac{1}{\pi}\int_{\omega_{p1}}^{\pi} \sin\left[\left(i - \frac{1}{2}\right)\omega\right] d\omega$$

$$= -\frac{\cos\left[\left(i - \frac{1}{2}\right)\omega_{p1}\right]}{\left(i - \frac{1}{2}\right)\pi}, \qquad 1 \le i \le \frac{N_1}{2}.$$

For type B systems the elements of Q are given by

$$q(i, j) = \frac{1}{\pi}\left\{\int_0^{\omega_{s1}} \cos(i\omega)\cos(j\omega)\, d\omega + \int_{\omega_{p1}}^{\pi} \cos(i\omega)\cos(j\omega)\, d\omega\right\}, \qquad 0 \le i, j \le \frac{N_1 - 1}{2}$$

$$= \begin{cases} \dfrac{\pi + \omega_{s1} - \omega_{p1}}{\pi}, & i = j = 0 \\[2ex] \dfrac{\pi + \omega_{s1} - \omega_{p1}}{2\pi} + \dfrac{\sin(2i\omega_{s1}) - \sin(2i\omega_{p1})}{4i\pi}, & i = j \neq 0 \\[2ex] \dfrac{\sin[(i - j)\omega_{s1}] - \sin[(i - j)\omega_{p1}]}{2(i - j)\pi} + \dfrac{\sin[(i + j)\omega_{s1}] - \sin[(i + j)\omega_{p1}]}{2(i + j)\pi}, & i \neq j \end{cases}$$

and the elements of $p^T = [p_0 \;\; p_1 \;\; \cdots \;\; p_{(N_1 - 1)/2}]$ are given by

$$p_i = -\frac{1}{\pi}\int_{\omega_{p1}}^{\pi} \cos(i\omega)\, d\omega, \qquad 0 \le i \le \frac{N_1 - 1}{2}$$

$$= \begin{cases} \dfrac{\omega_{p1} - \pi}{\pi}, & i = 0 \\[2ex] \dfrac{\sin(i\omega_{p1})}{i\pi}, & i \neq 0. \end{cases}$$

It is easy to reformulate the set of linear constraint equations in terms of y_1, yielding the following form:

$$C y_1 = m \tag{4}$$

where C is a known matrix, for a given $H_0(z)$. Therefore, our optimization problem for designing $H_1(z)$ becomes

$$\min \Phi = \tfrac{1}{2} y_1^T Q y_1 + p^T y_1 + d \qquad \text{subject to}$$

$$C y_1 = m.$$

Following the technique developed in [12], [13], we can solve this design problem in closed form by the method of Lagrange multipliers. The Lagrange multiplier vector is

$$\lambda = [\lambda_1 \;\; \lambda_2 \;\; \cdots \;\; \lambda_k]^T$$

and the Lagrangian function is

$$\Lambda(y_1, \lambda) = \tfrac{1}{2} y_1^T Q y_1 + p^T y_1 + d - \lambda^T(C y_1 - m).$$

Imposing the necessary and sufficient conditions for the solution

$$\nabla_{y_1}\Lambda = 0$$

$$\nabla_{\lambda}\Lambda = 0$$

we arrive at the following system of linear equations for the filter coefficient vector and Lagrange multiplier vector:

$$\begin{bmatrix} -Q & C^T \\ C & 0 \end{bmatrix} \begin{pmatrix} y_1 \\ \lambda \end{pmatrix} = \begin{pmatrix} p \\ m \end{pmatrix}. \tag{5}$$

The resulting closed-form solutions are

$$y_1 = Q^{-1}C^T(CQ^{-1}C^T)^{-1}m + Q^{-1}[C^T(CQ^{-1}C^T)^{-1}CQ^{-1} - I]p \tag{6}$$

$$\lambda = (CQ^{-1}C^T)^{-1}(m + CQ^{-1}p). \tag{7}$$

Example 2.1: A type B system with $N_0 = 23$ and $N_1 = 25$ was designed. We first designed the 23-tap low-pass filter $H_0(z)$ with passband edge frequency $\omega_{p0} = 0.5\pi$ and stopband edge frequency $\omega_{s0} = 0.6\pi$, using the eigenfilter approach [15]. We chose this approach so that $H_0(z)$ and $H_1(z)$ would both be designed according to a least squares criterion. Then, the coefficients of $H_0(z)$ were used as known variables to obtain the C matrix in (4). The Q matrix and p vector were easily calculated, given the band-edge frequencies ω_{s1} and ω_{p1}, which were chosen to be 0.4π and 0.6π. The coefficients of $H_1(z)$ were then obtained by the simple matrix computations in (6). The coefficients of $H_0(z)$ and $H_1(z)$ are shown in Table I. The magnitude response plots of $H_0(z)$ and $H_1(z)$ are shown in Fig. 2. The choice of the low-pass filter will affect the filtering performance of the resulting high-pass filter. Our computer simulations showed that by choosing a narrower transition bandwidth for the low-pass filter we can always obtain the high-pass filter with good frequency response. In fact, since the design of the low-pass filter involves only the computation of the eigenvectors for a matrix determined by ω_{s0} and ω_{p0}, and the design of the high-pass filter involves only the simple matrix computation in (6) determined by ω_{s1} and ω_{p1}, a computer program has been written in which we only need to adjust the values of ω_{s0}, ω_{p0}, ω_{s1}, and ω_{p1} for finding the appropriate filters. Our experiments showed that with only a few tries we can easily obtain the desired filters. Notice that we have 12 design parameters for designing the low-pass filter $H_0(z)$ and 13 design parameters for designing the high-pass filter $H_1(z)$. The number of perfect-reconstruction constraints, according to (2), is 12. This implies that by designing the low-pass filter first, we have only one degree of freedom left for designing the high-pass filter. However, with the help of the Lagrange-multiplier method we show here that even with only one degree of freedom we have been able to design good filters.

TABLE I
IMPULSE RESPONSES OF THE OPTIMIZED ANALYSIS FILTERS IN EXAMPLE 2.1

n	$h_0(n)$	$h_1(n)$
0	0.32022072941022D-02	0.20757249335743D-03
1	−0.16270808813591D-01	−0.10547013494721D-02
2	0.26026195430109D-02	0.16928183546876D-02
3	0.26218446566847D-01	−0.60446771109714D-02
4	−0.14024093618472D-01	−0.97791939000393D-03
5	−0.35139065156312D-01	0.16845089340686D-01
6	0.38375232701711D-01	−0.30423069328650D-02
7	0.44088905465945D-01	−0.35790690426760D-01
8	−0.88969771578456D-01	0.11756760495661D-01
9	−0.49113286303353D-01	0.92205949206770D-01
10	0.31303768685160D-00	−0.11746190124199D-01
11	0.55198385409394D-00	−0.31271750139945D-00
12		0.51134218298697D-00

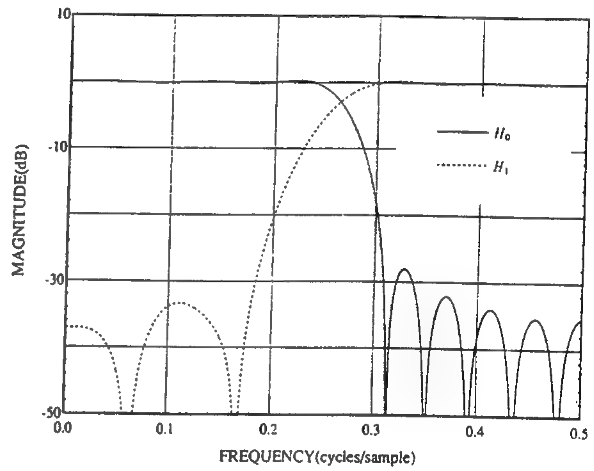


Fig. 2. Magnitude response plots for the analysis filters in example 2.1.

Example 2.2: A type B system with a larger difference in filter lengths, $N_0 = 15$ and $N_1 = 25$, was designed. The number of constraints for this system is 10. We have 13 design parameters, and thus 3 degrees of freedom, for designing the high-pass filter. The choice of the appropriate low-pass filter is done similar to that of example 2.1. The low-pass eigenfilter $H_0(z)$ with $\omega_{p0} = 0.46\pi$ and $\omega_{s0} = 0.6\pi$ was first designed. The high-pass filter $H_1(z)$ with $\omega_{s1} = 0.4\pi$ and $\omega_{p1} = 0.6\pi$ was then obtained using (6). The coefficients of $H_0(z)$ and $H_1(z)$ are shown in Table II. The magnitude response plots of $H_0(z)$ and $H_1(z)$ are shown in Fig. 3.

Example 2.3: In this example we designed a type A system with $N_0 = 16$ and $N_1 = 28$. The number of constraints is 12. Here we have 14 design parameters, and thus 2 degrees of freedom, for designing the high-pass filter. The low-pass eigenfilter $H_0(z)$ with $\omega_{p0} = 0.44\pi$ and $\omega_{s0} = 0.6\pi$ was first designed. The highpass filter $H_1(z)$ with $\omega_{s1} = 0.4\pi$ and $\omega_{p1} = 0.6\pi$ was then obtained. The coefficients of $H_0(z)$ and $H_1(z)$ are shown in Table III. The magnitude response plots of $H_0(z)$ and $H_1(z)$ are shown in Fig. 4.

In order to demonstrate the perfect-reconstruction property of the proposed Lagrange multiplier method, a computer simulation with double-precision arithmetic was run for a simple ramp input sequence $x(n)$. The reconstructed

TABLE II
IMPULSE RESPONSES OF THE OPTIMIZED ANALYSIS FILTERS IN EXAMPLE 2.2

n	$h_0(n)$	$h_1(n)$
0	0.19042472255677D-01	−0.18546886655979D-02
1	0.13764493664993D-01	−0.13406268915815D-02
2	−0.45029459441763D-01	0.73091436972806D-02
3	−0.22388494430843D-01	0.42936990344331D-02
4	0.91404228166495D-01	−0.26647270668886D-01
5	0.23010820706660D-01	−0.13507852518241D-01
6	−0.31583265276186D-00	0.24659762972978D-01
7	−0.52794281631871D-00	0.34620141498229D-01
8		−0.27144262652324D-01
9		−0.90014975943879D-01
10		0.33679245674349D-01
11		0.31042933163383D-00
12		−0.53108117920985D-00

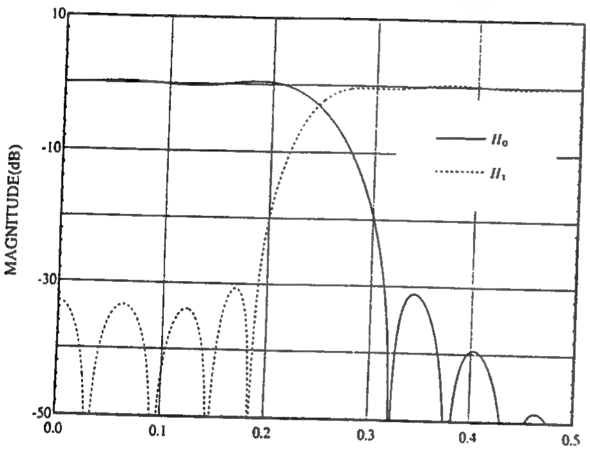


Fig. 3. Magnitude response plots for the analysis filters in example 2.2.

TABLE III
IMPULSE RESPONSES OF THE OPTIMIZED ANALYSIS FILTERS IN EXAMPLE 2.3

n	$h_0(n)$	$h_1(n)$
0	−0.50524665078789D-02	−0.19222322824304D-03
1	−0.22079439119735D-01	−0.84002161296320D-03
2	0.17886323630082D-01	0.12140454726470D-02
3	0.46720025466817D-01	0.41091254662707D-02
4	−0.41230804736905D-01	−0.56591175708268D-02
5	−0.92622557948810D-01	−0.18078827282842D-01
6	0.13353939880734D-00	0.13995252437161D-01
7	0.46283952040909D-00	0.33618364075630D-01
8		−0.34743037073021D-02
9		−0.56273810205632D-01
10		−0.24049238631185D-01
11		0.10878011285809D-00
12		0.11278611773976D-00
13		−0.47669441078655D-00

signal $\hat{x}(n)$ for examples 2.1, 2.2, and 2.3 are shown in Table IV.

III. LAGRANGE–NEWTON METHOD

While our first approach is simple and easy to use, it would probably be better for most situations to avoid the arbitrary choice of $H_0(z)$. Furthermore, we cannot use this approach for the design of equal-length filter banks, as the degree of freedom for designing $H_1(z)$ reduces to zero in

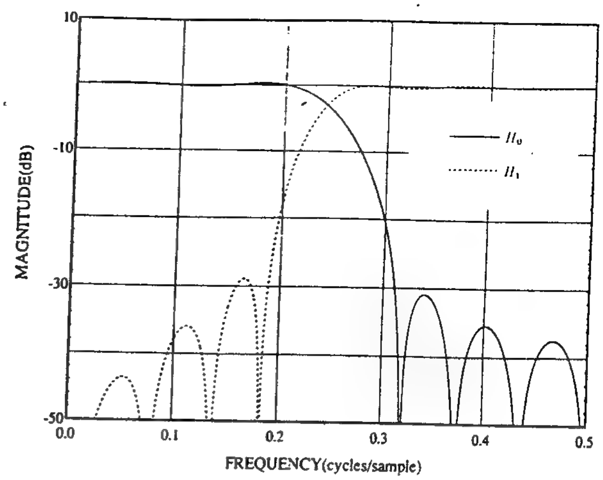


Fig. 4. Magnitude response plots for the analysis filters in example 2.3.

this case. Therefore, a systematic approach for finding the appropriate $H_0(z)$ and $H_1(z)$ simultaneously, in some optimal sense, and an approach which deals with the equal-length case is needed.

Our second approach, which we call the Lagrange-Newton method, meets all these requirements. Here, the impulse responses of $H_0(z)$ and $H_1(z)$ are treated simultaneously as unknowns. This makes (3) a set of nonlinear perfect-reconstruction constraint equations. Defining

$$a_0(n) = 2h_0\left(\frac{N_0}{2} - n\right), \qquad n = 1, 2, \cdots, \frac{N_0}{2}$$

and

$$b_0(n) = \begin{cases} h_0\left(\dfrac{N_0 - 1}{2}\right), & n = 0 \\ 2h_0\left(\dfrac{N_0 - 1}{2} - n\right), & n \neq 0 \end{cases}$$

we can express the zero-phase frequency response of $H_0(z)$ as a scalar product

$$H_0^*(e^{j\omega}) = y_0^T s_0(\omega)$$

where

$$y_0 = \begin{cases} \left[a_0(1) \quad a_0(2) \cdots a_0\left(\dfrac{N_0}{2}\right)\right]^T; \\ \text{type A} \\ \left[b_0(0) \quad b_0(1) \cdots b_0\left(\dfrac{N_0 - 1}{2}\right)\right]^T; \\ \text{type B} \end{cases}$$

and

$$s_0(\omega) = \begin{cases} \left[\cos\dfrac{\omega}{2} \quad \cos\dfrac{3\omega}{2} \cdots \cos\dfrac{N_0 - 1}{2}\omega\right]^T; \\ \text{type A} \\ \left[1 \quad \cos\omega \quad \cos 2\omega \cdots \cos\dfrac{N_0 - 1}{2}\omega\right]^T; \\ \text{type B.} \end{cases}$$

TABLE IV
A RAMP INPUT SEQUENCE $x(n)$ AND THE RECONSTRUCTED SIGNAL $\hat{x}(n)$ FOR EXAMPLE 2.1, EXAMPLE 2.2, AND EXAMPLE 2.3

n	$x(n)$	Example 2.1 $\hat{x}(n+23)$	Example 2.2 $\hat{x}(n+19)$	Example 2.3 $\hat{x}(n+21)$
0	1.0000000000000	1.0000000000000	1.0000000000000	1.0000000000000
1	2.0000000000000	2.0000000000000	2.0000000000000	2.0000000000000
2	3.0000000000000	3.0000000000000	3.0000000000000	3.0000000000000
3	4.0000000000000	4.0000000000000	4.0000000000000	4.0000000000000
4	5.0000000000000	5.0000000000000	5.0000000000000	5.0000000000000
5	6.0000000000000	6.0000000000000	6.0000000000000	6.0000000000000
6	7.0000000000000	7.0000000000000	7.0000000000000	7.0000000000000
7	8.0000000000000	8.0000000000000	8.0000000000000	8.0000000000000
8	9.0000000000000	9.0000000000000	9.0000000000000	9.0000000000000
9	10.0000000000000	10.0000000000000	10.0000000000000	10.0000000000000

The zero-phase frequency response of $H_1(z)$ can be expressed in the same form as that in Section II. There $H_0(z)$ and $H_1(z)$ are designed separately, which does not guarantee that the joint square error is minimal. Here we propose an approach which will minimize the following joint weighted square error:

$$\Phi = \frac{1}{2\pi}\left\{\alpha_0\left[\int_0^{\omega_{p0}} [1 - H_0^*(e^{j\omega})]^2\, d\omega + \alpha_{s0}\int_{\omega_{s0}}^{\pi} [H_0^*(e^{j\omega})]^2\, d\omega\right] + \alpha_1\left[\int_{\omega_{p1}}^{\pi} [1 - H_1^*(e^{j\omega})]^2\, d\omega + \alpha_{s1}\int_0^{\omega_{s1}} [H_1^*(e^{j\omega})]^2\, d\omega\right]\right\}$$

where α_{s0} and α_{s1} are the stopband weighting factors for $H_0(z)$ and $H_1(z)$, respectively, and α_0 and α_1 are the weighting factors for the whole approximation errors of $H_0(z)$ and $H_1(z)$, respectively. This objective function can be expressed as

$$\Phi = \tfrac{1}{2} y_0^T Q_0 y_0 + p_0^T y_0 + d_0 + \tfrac{1}{2} y_1^T Q_1 y_1 + p_1^T y_1 + d_1$$

where

$$Q_0 = \frac{\alpha_0}{\pi}\int_0^{\omega_{p0}} s_0(\omega) s_0^T(\omega)\, d\omega + \frac{\alpha_0\alpha_{s0}}{\pi}\int_{\omega_{s0}}^{\pi} s_0(\omega) s_0^T(\omega)\, d\omega$$

$$p_0^T = -\frac{\alpha_0}{\pi}\int_0^{\omega_{p0}} s_0^T(\omega)\, d\omega$$

$$d_0 = \frac{\alpha_0\omega_{p0}}{2\pi}$$

$$Q_1 = \frac{\alpha_1}{\pi}\int_{\omega_{p1}}^{\pi} s_1(\omega) s_1^T(\omega)\, d\omega + \frac{\alpha_1\alpha_{s1}}{\pi}\int_0^{\omega_{s1}} s_1(\omega) s_1^T(\omega)\, d\omega$$

$$p_1^T = -\frac{\alpha_1}{\pi}\int_{\omega_{p1}}^{\pi} s_1^T(\omega)\, d\omega$$

and

$$d_1 = \frac{\alpha_1(\pi - \omega_{p1})}{2\pi}.$$

For type A systems, the elements of Q_0, p_0, Q_1, and p_1 are given as follows:

$$q_0(i, j) = \frac{\alpha_0}{\pi}\left\{\int_0^{\omega_{p0}} \cos\left[\left(i - \frac{1}{2}\right)\omega\right]\cos\left[\left(j - \frac{1}{2}\right)\omega\right] d\omega + \int_{\omega_{s0}}^{\pi} \cos\left[\left(i - \frac{1}{2}\right)\omega\right]\cos\left[\left(j - \frac{1}{2}\right)\omega\right] d\omega\right\}$$

$$1 \le i, j \le \frac{N_0}{2}$$

$$= \begin{cases} \dfrac{\alpha_0}{\pi}\left\{\dfrac{\omega_{p0} + \alpha_{s0}(\pi - \omega_{s0})}{2} + \dfrac{\sin[(2i-1)\omega_{p0}] - \alpha_{s0}\sin[(2i-1)\omega_{s0}]}{4i - 2}\right\}, & i = j \\[2ex] \dfrac{\alpha_0}{\pi}\left\{\dfrac{\sin[(i-j)\omega_{p0}] - \alpha_{s0}\sin[(i-j)\omega_{s0}]}{2(i-j)} - \dfrac{\alpha_{s0}\sin[(i+j-1)\omega_{s0}] - \sin[(i+j-1)\omega_{p0}]}{2(i+j-1)}\right\}, & i \ne j. \end{cases}$$

$$p_{0,i} = -\frac{\alpha_0}{\pi}\int_0^{\omega_{p0}} \cos\left[\left(i - \frac{1}{2}\right)\omega\right] d\omega = -\frac{\alpha_0}{\pi}\left\{\frac{\sin\left[\left(i - \frac{1}{2}\right)\omega_{p0}\right]}{i - \frac{1}{2}}\right\}, \quad 1 \le i \le \frac{N_0}{2}.$$

$$q_1(i,j) = \frac{\alpha_1}{\pi}\left\{\alpha_{s1}\int_0^{\omega_{s1}} \sin\left[\left(i-\frac{1}{2}\right)\omega\right]\sin\left[\left(j-\frac{1}{2}\right)\omega\right]d\omega + \int_{\omega_{p1}}^{\pi}\sin\left[\left(i-\frac{1}{2}\right)\omega\right]\sin\left[\left(j-\frac{1}{2}\right)\omega\right]d\omega\right\}$$

$$1 \le i, j \le \frac{N_1}{2}$$

$$= \frac{\alpha_1}{\pi}\left\{\frac{\alpha_{s1}\omega_{s1} + \pi - \omega_{p1}}{2} + \frac{\sin[(2i-1)\omega_{p1}] - \alpha_{s1}\sin[(2i-1)\omega_{s1}]}{4i-2}\right\},$$

$$i = j$$

$$\frac{\alpha_1}{\pi}\left\{\frac{\alpha_{s1}\sin[(i-j)\omega_{s1}] - \sin[(i-j)\omega_{p1}]}{2(i-j)} - \frac{\alpha_{s1}\sin[(i+j-1)\omega_{s1}] - \sin[(i+j-1)\omega_{p1}]}{2(i+j-1)}\right\},$$

$$i \ne j.$$

$$p_{1,i} = -\frac{\alpha_1}{\pi}\int_{\omega_{p1}}^{\pi}\sin\left[\left(i-\frac{1}{2}\right)\omega\right]d\omega = -\frac{\alpha_1}{\pi}\left\{\frac{\cos\left[\left(i-\frac{1}{2}\right)\omega_{p1}\right]}{i-\frac{1}{2}}\right\}, \qquad 1 \le i \le \frac{N_1}{2}.$$

For type B systems, the elements of Q_0, p_0, Q_1, and p_1 are

$$q_0(i,j) = \frac{\alpha_0}{\pi}\left\{\int_0^{\omega_{p0}}\cos(i\omega)\cos(j\omega)\,d\omega + \alpha_{s0}\int_{\omega_{s0}}^{\pi}\cos(i\omega)\cos(j\omega)\,d\omega\right\}, \qquad 0 \le i, j \le \frac{N_0-1}{2}$$

$$= \begin{cases} \dfrac{\alpha_0(\alpha_{s0}\pi + \omega_{p0} - \alpha_{s0}\omega_{s0})}{\pi}, & i = j = 0 \\ \dfrac{\alpha_0}{\pi}\left[\dfrac{\omega_{p0} + \alpha_{s0}(\pi - \omega_{s0})}{2} + \dfrac{\sin(2i\omega_{p0}) - \alpha_{s0}\sin(2i\omega_{s0})}{4i}\right], & i = j \ne 0 \\ \dfrac{\alpha_0}{\pi}\left\{\dfrac{\sin[(i-j)\omega_{p0}] - \alpha_{s0}\sin[(i-j)\omega_{s0}]}{2(i-j)} + \dfrac{\sin[(i+j)\omega_{p0}] - \alpha_{s0}\sin[(i+j)\omega_{s0}]}{2(i+j)}\right\}, & i \ne j. \end{cases}$$

$$p_{0,i} = -\frac{\alpha_0}{\pi}\int_0^{\omega_{p0}}\cos(i\omega)\,d\omega, \qquad 0 \le i \le \frac{N_0-1}{2}$$

$$= \begin{cases} -\dfrac{\alpha_0\omega_{p0}}{\pi}, & i = 0 \\ -\dfrac{\alpha_0\sin(i\omega_{p0})}{i\pi}, & i \ne 0. \end{cases}$$

$$q_1(i,j) = \frac{\alpha_1}{\pi}\left\{\int_{\omega_{p1}}^{\pi}\cos(i\omega)\cos(j\omega)\,d\omega + \alpha_{s1}\int_0^{\omega_{s1}}\cos(i\omega)\cos(j\omega)\,d\omega\right\}, \qquad 0 \le i, j \le \frac{N_1-1}{2}$$

$$= \begin{cases} \dfrac{\alpha_1(\alpha_{s1}\omega_{s1} + \pi - \omega_{p1})}{\pi}, & i = j = 0 \\ \dfrac{\alpha_1}{\pi}\left[\dfrac{\alpha_{s1}\omega_{s1} + \pi - \omega_{p1}}{2} + \dfrac{\alpha_{s1}\sin(2i\omega_{s1}) - \sin(2i\omega_{p1})}{4i}\right], & i = j \ne 0 \\ \dfrac{\alpha_1}{\pi}\left\{\dfrac{\alpha_{s1}\sin[(i-j)\omega_{s1}] - \sin[(i-j)\omega_{p1}]}{2(i-j)} + \dfrac{\alpha_{s1}\sin[(i+j)\omega_{s1}] - \sin[(i+j)\omega_{p1}]}{2(i+j)}\right\}, & i \ne j. \end{cases}$$

$$p_{1,i} = -\frac{\alpha_1}{\pi}\int_{\omega_{p1}}^{\pi}\cos(i\omega)\,d\omega, \qquad 0 \le i \le \frac{N_1-1}{2}$$

$$= \begin{cases} -\dfrac{\alpha_1(\pi - \omega_{p1})}{\pi}, & i = 0 \\ \dfrac{\alpha_1\sin(i\omega_{p1})}{i\pi}, & i \ne 0. \end{cases}$$

Next, we can easily convert the set of nonlinear constraint equations (3) into the following form:

$$\begin{cases} u_l(y_0, y_1) = y_0^T D_l y_l = 0, & l = 1, 2, \cdots, k-1 \\ u_l(y_0, y_1) = y_0^T D_l y_l - \frac{1}{2} = 0, & l = k \end{cases}$$

where D_l can easily be determined when the lengths of the filters are given. Defining

$$y = [y_0^T \ \ y_1^T]^T$$

and

$$u(y) = [u_1(y) \ \ u_2(y) \ \cdots \ u_k(y)]^T$$

we can formulate our optimization problem as

$$\min \Phi(y) \qquad \text{subject to } u(y) = 0 \tag{8}$$

which is a nonlinear programming problem with nonlinear constraints.

This problem can be solved iteratively using the Lagrange–Newton method [16]. The Lagrange function is

$$L(y, \lambda) = \Phi(y) - \lambda^T u$$

where

$$\lambda = [\lambda_1 \ \ \lambda_2 \ \cdots \ \lambda_k]^T.$$

We define

$$\nabla = [\nabla_y^T \ \ \nabla_\lambda^T]^T.$$

Then, the condition for the stationary point y^*, λ^* is

$$\nabla L(y^*, \lambda^*) = 0.$$

Expanding ∇L in a Taylor series about $y^{(i)}$, $\lambda^{(i)}$ yields

$$\nabla L(y^{(i)} + \delta y, \lambda^{(i)} + \delta\lambda)$$
$$= \nabla L(y^{(i)}, \lambda^{(i)}) + [\nabla^2 L(y^{(i)}, \lambda^{(i)})] \begin{pmatrix} \delta y \\ \delta\lambda \end{pmatrix} + \cdots.$$

Neglecting higher order terms and setting the left-hand side to zero gives the iteration

$$[\nabla^2 L(y^{(i)}, \lambda^{(i)})] \begin{pmatrix} \delta y \\ \delta\lambda \end{pmatrix} = -\nabla L(y^{(i)}, \lambda^{(i)}).$$

This is solved to give corrections δy and $\delta\lambda$. Defining $\lambda^{(i+1)} = \lambda^{(i)} + \delta\lambda$, we then obtain the following system:

$$\begin{bmatrix} G^{(i)} & -A^{(i)} \\ -A^{(i)T} & 0 \end{bmatrix} \begin{pmatrix} \delta y \\ \lambda^{(i+1)} \end{pmatrix} = \begin{pmatrix} -g^{(i)} \\ u^{(i)} \end{pmatrix} \tag{9}$$

where

$$G^{(i)} = \nabla_y^2 L^{(i)}$$
$$A^{(i)} = [\nabla u_1^{(i)} \ \ \nabla u_2^{(i)} \ \cdots \ \nabla u_k^{(i)}]$$
$$g^{(i)} = \nabla_y \Phi^{(i)}$$

and where the superscript (i) represents that the values are evaluated at the ith iteration $y^{(i)}$ and $\lambda^{(i)}$. Notice that the linear system of (9) has the same form as (5) and can readily be solved, giving

$$\delta y = -G^{-1}A(A^TG^{-1}A)^{-1}u + G^{-1}[A(A^TG^{-1}A)^{-1}A^TG^{-1} - I]g \tag{10}$$

$$\lambda^{(i+1)} = (A^TG^{-1}A)^{-1}(A^TG^{-1}g - u). \tag{11}$$

The analytical forms for A, G, and g can easily be derived as follows:

$$A = \begin{bmatrix} D_1(1r)y_1 & D_2(1r)y_1 & \cdots & D_k(1r)y_1 \\ D_1(2r)y_1 & D_2(2r)y_1 & \cdots & D_k(2r)y_1 \\ \vdots & \vdots & \vdots & \vdots \\ D_1(Nr)y_1 & D_2(Nr)y_1 & \cdots & D_k(Nr)y_1 \\ y_0^TD_1(1c) & y_0^TD_2(1c) & \cdots & y_0^TD_k(1c) \\ y_0^TD_1(2c) & y_0^TD_2(2c) & \cdots & y_0^TD_k(2c) \\ \vdots & \vdots & \vdots & \vdots \\ y_0^TD_1(Mc) & y_0^TD_2(Mc) & \cdots & y_0^TD_k(Mc) \end{bmatrix}$$

where $D_k(Nr)$ and $D_k(Mc)$ represent the Nth row and the Mth column of D_k, respectively, and

$$M = \begin{cases} \dfrac{N_1}{2}, & \text{for type A} \\ \dfrac{N_1 + 1}{2}, & \text{for type B} \end{cases}$$

and

$$N = \begin{cases} \dfrac{N_0}{2}, & \text{for type A} \\ \dfrac{N_0 + 1}{2}, & \text{for type B} \end{cases}$$

$$G = \begin{bmatrix} Q_0 & -\sum_{i=1}^{k} \lambda_i D_i \\ -\sum_{i=1}^{k} \lambda_i D_i^T & Q_1 \end{bmatrix}$$

$$g = \begin{pmatrix} Q_0 y_0 + p_0 \\ Q_1 y_1 + p_1 \end{pmatrix}.$$

Therefore, we simply form A, G, g, and u, and use (10) and (11) to find δy and $\lambda^{(i+1)}$. Then, $y^{(i+1)}$ is given by

$$y^{(i+1)} = y^{(i)} + \delta y. \tag{12}$$

Example 3.1: A type B system with $N_0 = 23$, $N_1 = 25$, $\omega_{p0} = \omega_{s1} = 0.4\pi$, and $\omega_{p1} = \omega_{s0} = 0.6\pi$ was designed. Our Lagrange–Newton method requires initial approximations $y^{(1)}$ and $\lambda^{(1)}$, and uses (10)–(12) to generate the iterative sequence $\{y^{(i)}, \lambda^{(i)}\}$. As with most nonlinear optimization problems, our computer simulations showed that the solution was sensitive to the initial approximations. However for unequal-length filter banks, our first method, the Lagrange multiplier method, served

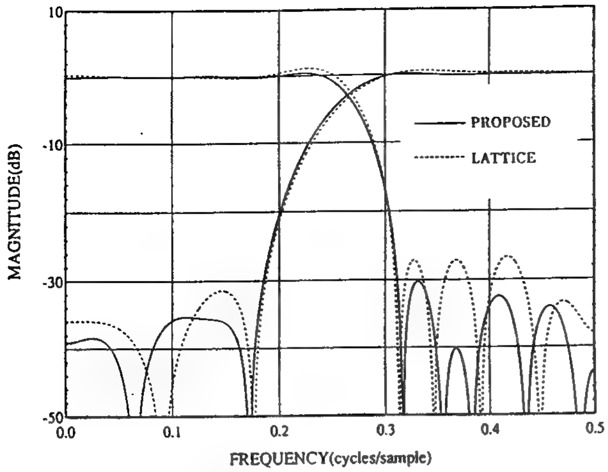


Fig. 5. Magnitude response plots of the proposed approach and the lattice approach in example 3.1.

TABLE V
IMPULSE RESPONSES OF THE OPTIMIZED ANALYSIS FILTERS IN EXAMPLE 3.1

n	$h_0(n)$	$h_1(n)$
0	0.19664310885798D-02	0.26030504425555D-03
1	−0.15071897603198D-01	−0.19951327027936D-02
2	0.39538725460162D-02	0.14764582380207D-02
3	0.24878605633241D-01	−0.40115824373647D-02
4	−0.14200153852088D-01	−0.89182016597635D-03
5	−0.36006015487364D-01	0.14407416308426D-01
6	0.35461028533182D-01	−0.23455316659266D-02
7	0.47745012128669D-01	−0.35978610240721D-01
8	−0.89925971104091D-01	0.10662107010783D-01
9	−0.53096857923338D-01	0.91238511647933D-01
10	0.31042847037014D-00	−0.87734159524355D-02
11	0.55179158942662D-00	−0.31495941897510D-00
12		0.51290303596677D-00

as an easy way to approximate the initial estimates. We simply used the results of example 2.1 as the initial approximations: 1) We designed a 23-tap low-pass eigenfilter $H_0(z)$ with $\omega_{p0} = 0.5\pi$ and $\omega_{s0} = 0.6\pi$ to get $y_0^{(1)}$. 2) We used (6) and (7) to find $y_1^{(1)}$ and $\lambda^{(1)}$, respectively.

Then, $y^{(1)} = [y_0^{(1)T} \; y_1^{(1)T}]^T$ and $\lambda^{(1)}$ were used iteratively to find the optimal solution. With $\alpha_0 = \alpha_1 = \alpha_{s0} = \alpha_{s1} = 1$, our algorithm converged to the solution within 11 iterations. The magnitude response plots of $H_0(z)$ and $H_1(z)$ are shown in Fig. 5. The coefficient of $H_0(z)$ and $H_1(z)$ are shown in Table V. To compare with other results reported recently, the magnitude response plots of the lattice approach of [11] are also shown in Fig. 5, and the peak ripples in the passband and stopband are summarized in Table VI. It is evident that the proposed approach has smaller peak ripples.

Example 3.2: A type A system with $N_0 = N_1 = 22$, $\omega_{p0} = \omega_{s1} = 0.4\pi$, and $\omega_{s0} = \omega_{p1} = 0.6\pi$ was designed. For such an equal-length system, our computer experiments showed that the JMSE filters [17] served as good candidates for the initial approximations. These filters were designed by approximating the ideal brick-wall half-band filters using the downhill simplex method [18]. We simply used the computer program in [17] to obtain our initial approximations for $y^{(1)}$, and set $\lambda^{(1)} = 0$. The initial magnitude response plots are shown in Fig. 6. With $\alpha_0 = 1$, $\alpha_1 = 2$, $\alpha_{s0} = 1$, and $\alpha_{s1} = 0.8$ the solution was obtained within 8 iterations. Here, by adjusting the weighting factors, various filter performance criteria can be accommodated. For the chosen weighting factors, we were able to obtain better filtering performance than [11]. The magnitude response plots of $H_0(z)$ and $H_1(z)$ are shown in Fig. 7. The coefficients of $H_0(z)$ and $H_1(z)$ are

TABLE VI
COMPARISON BETWEEN THE PROPOSED APPROACH AND THE LATTICE APPROACH FOR EXAMPLE 3.1. HERE δ_1 AND δ_2 DENOTE THE PEAK-RIPPLE SIZES IN THE PASSBAND AND STOPBAND, RESPECTIVELY

	Lattice Approach		Proposed Approach		$\delta_{ratio} = \frac{\delta_{lattice}}{\delta_{proposed}}$	
	H_0	H_1	H_0	H_1	H_0	H_1
δ_1	0.0327	0.0349	0.0224	0.0230	1.46	1.51
δ_2	0.0449	0.0267	0.0307	0.0171	1.46	1.56

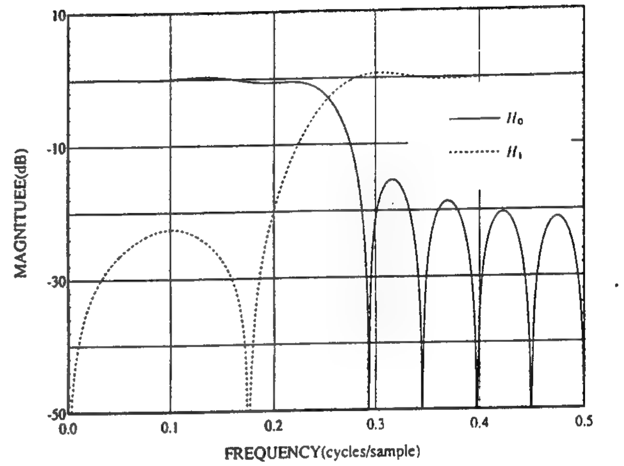


Fig. 6. Magnitude response plots for the initial analysis filters in example 3.2.

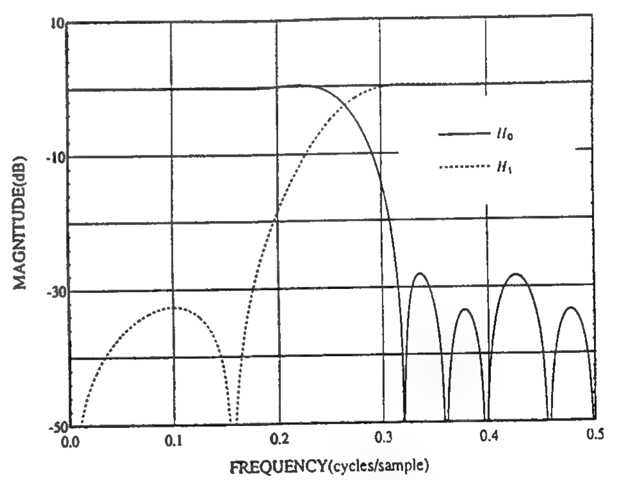


Fig. 7. Magnitude response plots for the analysis filters in example 3.2.

TABLE VII
IMPULSE RESPONSES OF THE OPTIMIZED ANALYSIS FILTERS IN EXAMPLE 3.2

n	$h_0(n)$	$h_1(n)$
0	0.13214141754298D-02	0.24489011491443D-03
1	−0.13588732555019D-01	−0.25183219151237D-02
2	0.16361001428450D-01	0.24872587441326D-02
3	0.14471935841032D-01	0.82847670444218D-02
4	−0.33083436288235D-01	−0.11927299751857D-01
5	−0.10335603320186D-01	−0.17648198575570D-01
6	0.60231455220816D-01	0.33963707314799D-01
7	−0.93183731576413D-02	0.39646017249692D-01
8	−0.11708023337247D-00	−0.90292313119252D-01
9	0.10149306240288D-00	−0.13843420538781D-00
10	0.48371640962495D-00	0.46073324704670D-00

TABLE VIII
COMPARISON BETWEEN THE PROPOSED APPROACH AND THE LATTICE APPROACH FOR EXAMPLE 3.2. HERE δ_1 AND δ_2 DENOTE THE PEAK-RIPPLE SIZES IN THE PASSBAND AND STOPBAND, RESPECTIVELY

	Lattice Approach		Proposed Approach		$\delta_{\text{ratio}} = \frac{\delta_{\text{lattice}}}{\delta_{\text{proposed}}}$	
	H_0	H_1	H_0	H_1	H_0	H_1
δ_1	0.0246	0.0260	0.0133	0.0252	1.85	1.03
δ_2	0.0592	0.0307	0.0400	0.0234	1.48	1.31

TABLE IX
A RAMP INPUT SEQUENCE $x(n)$ AND THE RECONSTRUCTED SIGNAL $\hat{x}(n)$ FOR EXAMPLE 3.1 AND EXAMPLE 3.2

n	$x(n)$	Example 3.1 $\hat{x}(n+23)$	Example 3.2 $\hat{x}(n+21)$
0	1.0000000000000	1.0000000000000	1.0000000000000
1	2.0000000000000	2.0000000000000	2.0000000000000
2	3.0000000000000	3.0000000000000	3.0000000000000
3	4.0000000000000	4.0000000000000	4.0000000000000
4	5.0000000000000	5.0000000000000	5.0000000000000
5	6.0000000000000	6.0000000000000	6.0000000000000
6	7.0000000000000	7.0000000000000	7.0000000000000
7	8.0000000000000	8.0000000000000	8.0000000000000
8	9.0000000000000	9.0000000000000	9.0000000000000
9	10.0000000000000	10.0000000000000	10.0000000000000

shown in Table VII. The comparison of peak ripples with [11] is summarized in Table VIII.

In order to demonstrate the perfect-reconstruction property of the proposed approach, a computer simulation with double-precision arithmetic was run for a simple ramp input sequence $x(n)$. The reconstructed signal $\hat{x}(n)$ for examples 3.1 and 3.2 are shown in Table IX.

IV. CONCLUSIONS

We have presented two new approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks. Using these Lagrange multiplier approaches, we have been able to formulate the design problem first as a quadratic programming problem with linear constraints, and then as one with nonlinear constraints. Closed-form solutions for the first approach, and for the iterative problem in the second approach have been derived. Several design examples have been given to show the effectiveness of the proposed approaches. When compared to other results recently reported, the proposed approaches appear to have better filtering performance. One further observation about the first approach is that, when the optimal infinite-precision impulse response of $h_0(n)$ is rounded to the nearest power-of-two coefficients, we can still obtain the impulse response of $h_1(n)$ by using (6), and we thus obtain a perfect-reconstruction system with low-complexity $H_0(z)$.

REFERENCES

[1] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[2] J. Woods and S. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1278–1288, Oct. 1986.

[3] M. J. T. Smith and T. P. Barnwell, III, "Exact reconstruction techniques for tree structured subband coders," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 434–441, June 1986.

[4] M. Vetterli, "Filter banks allowing perfect reconstruction," *Signal Processing*, vol. 10, pp. 219–244, Apr. 1986.

[5] P. P. Vaidyanathan, "Theory and design of M channel maximally decimated quadrature mirror filters with arbitrary M, having perfect reconstruction property," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 476–492, Apr. 1987.

[6] K. Nayebi, T. P. Barnwell, III, and M. J. T. Smith, "The time domain analysis and design of exactly reconstructing FIR analysis/synthesis filter banks," in *Proc. Int. Conf. ASSP*, 1990, pp. 1735–1738.

[7] M. J. T. Smith and S. L. Eddins, "Subband coding of images with octave band tree structures," in *Proc. Int. Conf. ASSP*, 1987, pp. 1382–1385.

[8] M. J. T. Smith and S. L. Eddins, "Analysis/synthesis techniques for subband image coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 1446–1456, Aug. 1990.

[9] M. Vetterli, "A theory of multirate filter banks," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-35, pp. 356–372, Mar. 1987.

[10] M. Vetterli and D. Le Gall, "Perfect reconstruction FIR filter banks: Some properties and factorizations," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1057–1071, July 1989.

[11] T. Q. Nguyen and P. P. Vaidyanathan, "Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 676–690, May 1989.

[12] G. W. Medlin, J. W. Adams, and C. T. Leondes, "Lagrange multiplier approach to the design of FIR filters for multirate applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1210–1219, Oct. 1988.

[13] G. W. Medlin and J. W. Adams, "A new technique for maximally linear differentiators," in *Proc. Int. Conf. ASSP*, 1989, pp. 825–828.

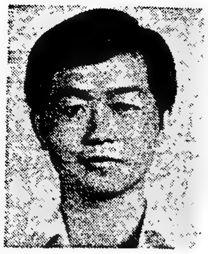
[14] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[15] P. P. Vaidyanathan and T. Q. Nguyen, "Eigenfilters: A new approach to least squares FIR filter design and applications including Nyquist filters," *IEEE Trans. Circuits Syst.*, vol. CAS-34, pp. 11–23, Jan. 1987.

[16] R. Fletcher, *Practical Methods of Optimization*, vol. 2. New York: Wiley, 1981.

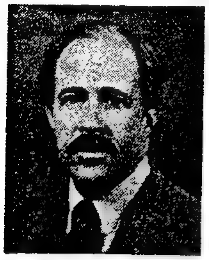
[17] J. C. Darragh, "Subband and transform coding of images," Ph.D. dissertation, Elec. Eng. Dep., Univ. California, Los Angeles, CA, 1989.

[18] W. K. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge: Cambridge University Press, 1988.



Bor-Rong Horng (S'86–M'90) received the B.S. and M.S. degrees in power mechanical engineering from National Tsing-Hua University, Taiwan, in 1980 and 1982, and the M.S., Engr., and Ph.D. degrees in electrical engineering from the University of California, Los Angeles (UCLA) in 1985, 1988, and 1990, respectively.

From 1982 to 1984 he was a Member of the Technical Staff at the Aeronautical Research Laboratory, Taiwan. From 1985 to 1989 he was a Teaching Assistant, and from 1989 to 1990 he was a Research Associate, in the Department of Electrical Engineering at UCLA. Since 1990 he has been with the Digital Communications Division of Rockwell International, Newport Beach, CA, where he is engaged in the design of telecommunication integrated circuits. His research interests include digital filter design, data compression, and integrated circuits for telecommunication applications.



Alan N. Willson, Jr. (S'66–M'67–SM'73–F'78) received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, GA, in 1961, and the M.S. and Ph.D. degrees from Syracuse University, Syracuse, NY, in 1965 and 1967, respectively.

From 1961 to 1964 he was with IBM, Poughkeepsie, NY. He was an Instructor in Electrical Engineering at Syracuse University from 1965 to 1967. From 1967 to 1973 he was a Member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. Since 1973 he has been on the faculty of the University of California, Los Angeles, where he is now Professor of Engineering and Applied Science in the Electrical Engineering Department. In addition, he served the UCLA School of Engineering and Applied Science as Assistant Dean for Graduate Studies, from 1977 through 1981, and is currently Associate Dean of Engineering. He has been engaged in research concerning computer-aided circuit analysis and design, the stability of distributed circuits, properties of nonlinear networks, theory of active circuits, digital signal processing, analog circuit fault diagnosis, and integrated circuits for signal processing. He is editor of the book *Nonlinear Networks: Theory and Analysis* (IEEE Press, 1974).

Dr. Willson is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, the Society of Industrial and Applied Mathematics, and the American Society for Engineering Education. From 1977 to 1979 he served as Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS. In 1980 he was General Chairman of the Fourteenth Asilomar Conference on Circuits, Systems, and Computers. During 1984 he served as President of the IEEE Circuits and Systems Society. He was the recipient of the 1978 Guillemin–Cauer Award of the IEEE Circuits and Systems Society, the 1982 George Westinghouse Award of the American Society for Engineering Education, the 1982 Distinguished Faculty Award of the UCLA Engineering Alumni Association, the 1984 Myril B. Reed Best Paper Award of the Midwest Symposium on Circuits and Systems, and the 1985 W. R. G. Baker Award of the IEEE.

The Design of Low-Complexity Linear-Phase FIR Filter Banks Using Powers-of-Two Coefficients with an Application to Subband Image Coding

Bor-Rong Horng, *Member, IEEE,* Henry Samueli, *Member, IEEE,* and Alan N. Willson, Jr., *Fellow, IEEE*

Abstract—**An optimization technique is presented for the design of multiplierless two-channel linear-phase finite-duration impulse-response (FIR) filter banks. It is shown to yield filter banks with good filtering performance and nearly perfect signal reconstruction. The design employs filters whose coefficients are represented by a canonic signed-digit (CSD) code. When applied to subband image coding this technique provides an easy way to design low-complexity analysis/synthesis filter banks for high-performance codecs. Examples concerning filter design and the application of such filters to subband image coding are given.**

I. Introduction

Subband image coding has recently been shown to be an effective technique for image compression [1]–[4]. Although this technique can yield high-quality coding systems at low bit rates, it generally requires the implementation of sophisticated analysis/synthesis filter banks, which increases system complexity. The filter bank's operation requires numerous multiplications and additions. Multiplication, in particular, is extremely time consuming. With current advanced very-large-scale integration (VLSI) technologies, fast multipliers (operating at speeds exceeding 100 MHz [5]) are available. Such multipliers employ highly parallel processing, which requires a large chip area. If filter banks were employed in high-speed applications such as real-time image compression systems, a separate fast multiplier would probably be required for each filter coefficient, which would surely be unacceptable from a hardware-complexity point of view. However, if a multiplication operation could be replaced by only a few additions or subtractions then the complexity of the entire analysis/synthesis filter bank would be reduced quite dramatically to a point where its implementation in a fast real-time system becomes feasible. In this paper we show how such a goal can be achieved. We employ a discrete coefficient optimization technique to design two-channel linear-phase finite-duration impulse-response (FIR) filter banks that exhibit good filtering performance and nearly perfect signal reconstruction while requiring far less hardware complexity in comparison to conventional filter banks using floating-point coefficients [6]–[8].

We represent our filter coefficients by a radix-2 canonic signed-digit (CSD) code [9]. By adding the flexibility of negative digits to a conventional binary code, the radix-2 signed-digit representation of a fractional number c is given by

$$c = \sum_{k=1}^{L} s_k 2^{-p_k} \tag{1}$$

where $s_k \in \{-1, 0, 1\}$ and $p_k \in \{0, 1, \cdots, M\}$. The number representation specified by (1) has $M + 1$ total (ternary) digits and L nonzero digits. The number of adders/subtractors required to realize such a coefficient is $L - 1$, one less than the number of nonzero digits. In general there are several signed-digit representations for a given number. The CSD code is that representation with the minimum number of nonzero digits and for which no two nonzero digits s_k are adjacent. A well-known feature of the CSD code is its ability to represent most numbers with many fewer nonzero digits. For example, the 8-bit two's complement representation of $127/128 = 0.9921875$ has seven nonzero digits (0.1111111) whereas the eight-digit radix-2 CSD representation of the same number has only two nonzero ternary digits as given by $1.000000\underline{1}$, where $\underline{1}$ denotes -1. Thus, only a single subtractor would be required to implement a multiplier with a coefficient having this value. It is this feature of the CSD code that makes it possible to design low-complexity high-performance filter banks suitable for single-chip VLSI implementations.

II. Two-Channel Filter Banks

Subband image coding involves the design of two-dimensional filter banks. In the present work we restrict our attention to the simple case of a two-channel system in one dimension. Such filter banks can be cascaded in a tree structure to provide an arbitrarily fine division of the signal, in frequency, and can be directly applied to two-dimensional subband image coding systems by using separable filter banks [10] which first perform the filtering on the rows and then on the columns of an image.

A generic two-channel FIR filter bank is shown in Fig. 1. Here $H_0(z)$ and $H_1(z)$ represent the lowpass and highpass

Manuscript received February 4, 1991; revised September 27, 1991. This paper was recommended by Associate Editor John W. Woods. This work was supported by the Office of Naval Research, under Grant N00014-91-J-1852. This paper was recommended by Associate Editor John W. Woods.

B.-R. Horng was with the Electrical Engineering Department, University of California, Los Angeles, CA 90024-1594. He is now with the Digital Communications Division, Rockwell International, Newport Beach, CA 92658-8902.

H. Samueli and A. N. Willson are with the Electrical Engineering Department, University of California, Los Angeles 90024-1594.

IEEE Log Number 9104586.

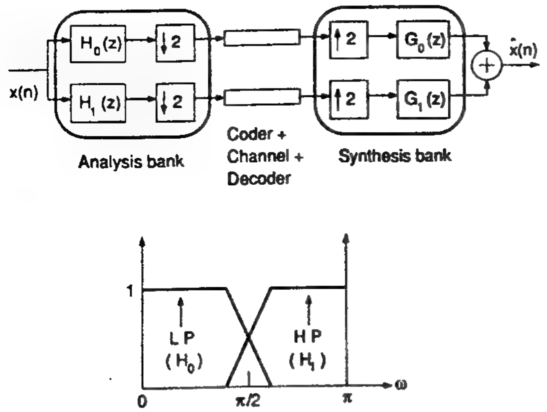


Fig. 1. Two-channel analysis/synthesis filter bank system.

filters, respectively, in the analysis bank, and $G_0(z)$ and $G_1(z)$ are the synthesis filters. Assuming perfect channels and codecs, it is well known that the reconstructed signal $\hat{x}(n)$ can be related to the input signal $x(n)$ by

$$\hat{X}(z) = \tfrac{1}{2}\left[H_0(z)G_0(z) + H_1(z)G_1(z)\right]X(z) + \tfrac{1}{2}\left[H_0(-z)G_0(z) + H_1(-z)G_1(z)\right]X(-z).$$

Furthermore, by choosing

$$G_0(z) = 2H_1(-z)$$
$$G_1(z) = -2H_0(-z)$$

we have

$$\hat{X}(z) = \left[H_0(z)H_1(-z) - H_0(-z)H_1(z)\right]X(z). \quad (2)$$

If we impose the following pure-delay constraint

$$H_0(z)H_1(-z) - H_1(z)H_0(-z) = z^{-2k+1}. \quad (3)$$

then

$$\hat{X}(z) = z^{-2k+1}X(z).$$

Thus, we obtain a perfect reconstruction system where the output $\hat{x}(n)$ is a delayed replica of the input $x(n)$. In some applications linear-phase filters are preferred over nonlinear-phase filters for image coding [11]. As described in [8], by combining the linear-phase constraint with the pure-delay constraint there are, in total, 16 possible types of $H_0(z)$, $H_1(z)$ pairs to consider, only two of which yield nontrivial analysis filters:

1) Both filters have even length and opposite symmetry, denoted in [8] as type A systems.
2) Both filters have odd length and are symmetric, denoted in [8] as type B systems.

Using terminology defined in [12] there are four cases of linear-phase FIR filters, depending on whether the filter length is odd or even and whether the impulse response is symmetrical or antisymmetrical:

Case 1) The impulse response is symmetrical and the filter length is odd.
Case 2) The impulse response is symmetrical and the filter length is even.
Case 3) The impulse response is antisymmetrical and the filter length is odd.
Case 4) The impulse response is antisymmetrical and the filter length is even.

Then, for type A systems the analysis filters are either case 2 or case 4 since the lengths are even. It can easily be shown that case 2 requires $H_1(e^{j\pi}) = 0$ and thus it cannot realize a highpass filter. Similarly, case 4 cannot realize a lowpass filter. Therefore $H_1(z)$ must be case 4 and $H_0(z)$ must be case 2.

For type B systems, it is obvious that both $H_0(z)$ and $H_1(z)$ must be case 1. Furthermore, by examining the pure-delay constraint (3), and by considering the coefficient symmetry/antisymmetry of linear-phase filters, we can make the following observations (assuming the lengths of $h_0(n)$ and $h_1(n)$ to be N_0 and N_1, respectively):

1) The sum of the lengths must be a multiple of 4 [8].
2) The number of independent constraint equations in (3) reduces to $(N_0 + N_1)/4$.
3) The constraint equations can be expressed as

$$\tfrac{1}{2}\delta\left(i - \frac{N_0 + N_1}{4}\right) = \sum_{k=0}^{2i-1} (-1)^k h_0(2i - 1 - k)h_1(k), \quad i = 1, 2, \cdots, \frac{N_0 + N_1}{4}. \quad (4)$$

It should be noted here that for type B systems $h_0(n) = 0$ for $n \geq N_0$ and $h_1(n) = 0$ for $n \geq N_1$. By adding the coefficient symmetry/antisymmetry of the linear-phase filters (4) can be expressed in the matrix form

$$\tilde{C}\tilde{y}_1 = m \quad (5)$$

where $\tilde{y}_1$ is an $(l_1 + 1)$-dimensional column vector

$$\tilde{y}_1 = \left[h_1(0)h_1(1) \cdots h_1(l_1)\right]^T$$

m is an $(N_0 + N_1)/4$-dimensional column vector

$$m = \left[0 \cdots 0 \ \tfrac{1}{2}\right]^T$$

$\tilde{C}$ is an $(N_0 + N_1)/4$-by-$(l_1 + 1)$ matrix with the elements formed by $h_0(n)$, $n = 0, 1, \cdots, l_0$ and

$$\begin{cases} l_0 = \dfrac{N_0}{2} - 1 & l_1 = \dfrac{N_1}{2} - 1, \quad \text{type } A \\ l_0 = \dfrac{N_0 - 1}{2} & l_1 = \dfrac{N_1 - 1}{2}, \quad \text{type } B. \end{cases}$$

These constraint equations are nonlinear because both $h_0(n)$ and $h_1(n)$ are involved. However, if the lowpass impulse response $h_0(n)$ is given, they then become linear. It has been pointed out in [13] that if only the coefficients of the lowpass filter $H_0(z)$ are restricted to CSD coefficients then good perfect-reconstruction systems can be obtained. The design procedure is given as follows:

1) The lowpass impulse response $h_0(n)$ is obtained by using the Lagrange–Newton method in [6]. It is rounded

to the nearest CSD code, with the number of nonzero digits as low as possible. This assures the low complexity of the lowpass filter.

2) Depending on whether or not the filter lengths are equal, the highpass filter $H_1(z)$ is obtained by:
   (a) The Unequal-Length Case: The rounded lowpass impulse response is used in the Lagrange multiplier method in [6] to obtain the highpass impulse response.
   (b) The Equal-Length Case: The rounded lowpass impulse response is used to obtain the $\tilde{C}$ in (5), and then the highpass impulse response is obtained by

$$\tilde{y}_1 = \tilde{C}^{-1}m. \qquad (6)$$

The perfect-reconstruction filter banks so obtained would require multipliers only for the implementation of $H_1(z)$ because each coefficient of $H_0(z)$ could be implemented by using only a few adders or subtractors. We then need only concentrate on searching for a suitable set of highpass CSD coefficients to achieve a totally multiplierless design. This procedure serves to provide a good starting point for the CSD search technique described in the next section.

III. The Optimization Algorithm

The main core of our proposed discrete optimization algorithm is the two-stage local search strategy recently reported [14]:

Stage 1) We search for the optimal scale factor, given L and M in (1), and we assign one more nonzero digit to those coefficients whose magnitude exceeds $\frac{1}{2}$, such that an appropriate objective function is minimized.

Stage 2) We use a bivariate local search technique [15] to find the best set of CSD coefficients, in the neighborhood of the scaled and rounded coefficients, which minimizes the objective function.

This two-stage local search strategy has been shown to be very efficient for finding a nearly optimal set of CSD coefficients in unconstrained FIR filter design [14]. Therefore we adopt this strategy and modify it to fit the needs of our filter-bank design problem.

When the filter bank coefficients are restricted to a relatively sparse set of coefficients such as the CSD coefficients, the constraint equations (4), which embody the perfect-reconstruction property, generally will not be satisfied. Therefore, we must establish an objective function that will yield good filtering performance while adhering to (4) as closely as possible. A reasonable objective function would be a joint weighted function of these two requirements. However, our computer simulations have shown that the constraint imposed by (4) is considerably more dominant than that of the filtering requirement. Thus, the objective function to be minimized is chosen as

$$\delta = \max_{0 \le \omega \le \pi} \left| \{ | [H_0(e^{j\omega}) H_1(-e^{j\omega}) - H_0(-e^{j\omega}) H_1(e^{j\omega})] | - 1.0 \} \right| \qquad (7)$$

which is the peak ripple of the signal-reconstruction error.

Our proposed CSD optimization algorithm is described as follows:

1) We employ the coefficients obtained by using the procedure in Section II as the starting point.
2) The two-stage local search strategy is then adopted to search for the optimal set of CSD coefficients for $H_1(z)$ such that (7) is minimized.
3) Finally, a scale factor (SF) is needed to scale the signal level back such that the overall system transfer function is close to 1. This scale factor, which is also rounded to the nearest CSD code, can be inserted right after the output of $H_1(z)$ and $G_0(z)$.

Two design examples are now given to illustrate the proposed technique.

Example 1: An equal-length case for type A systems is designed, where both $H_0(z)$ and $H_1(z)$ have 22 taps. The infinite-precision coefficients resulting from the Lagrange–Newton method in [6] are used as the starting point. $L = 2$ and $M = 16$ are chosen for $H_0(z)$, and $L = 4$ and $M = 16$ are chosen for $H_1(z)$. The resulting CSD coefficients are shown in Table I. The total number of adders/subtractors, including the scale factor, to implement the entire analysis filter bank is 85. Recently there have been several high-speed VLSI single-chip implementations of CSD FIR filters [16], [17]. In [17] a 64-tap CSD FIR linear-phase filter, working at video rate, has been implemented on a single chip. These results show that it is feasible to implement our CSD filter bank on a single chip using modern VLSI technology since our filter-bank complexity is less complicated than that of the 64-tap filter.

Fig. 2 shows the magnitude response plots of the infinite-precision optimal system and the CSD optimal system. As we can see, the filtering performance of the CSD design is almost as good as that of the infinite-precision design. The hardware complexity, however, has been reduced significantly. The price paid for the CSD design is the loss of perfect signal reconstruction. The overall system magnitude response of the CSD design is shown in Fig. 3. Here we see that the reconstruction error of the CSD design is less than 0.00026 dB. Such an extremely small reconstruction error is believed to be negligible in practice. This belief is also supported by the subband image coding experiment described in the next section.

Example 2: An unequal-length case for type B systems is designed. Here, $H_0(z)$ has 23 taps and $H_1(z)$ has 25 taps. Again, the starting point is obtained from the use of the Lagrange–Newton method [6]. Again, $L = 2$ and $M = 16$ are chosen for $H_0(z)$, and $L = 4$ and $M = 16$ are chosen for $H_1(z)$. Fig. 4 shows the resulting magnitude response plots of the infinite-precision optimal system and the CSD optimal system. Fig. 5 shows the overall system magnitude response of the CSD design. Again, we observe that the filtering performance of the CSD design is almost as good as that of the infinite-precision design. The reconstruction error is less than 0.00013 dB, which we also believe to be negligible in practice. The resulting CSD coefficients are shown in Table II. The total number of adders/subtractors, including

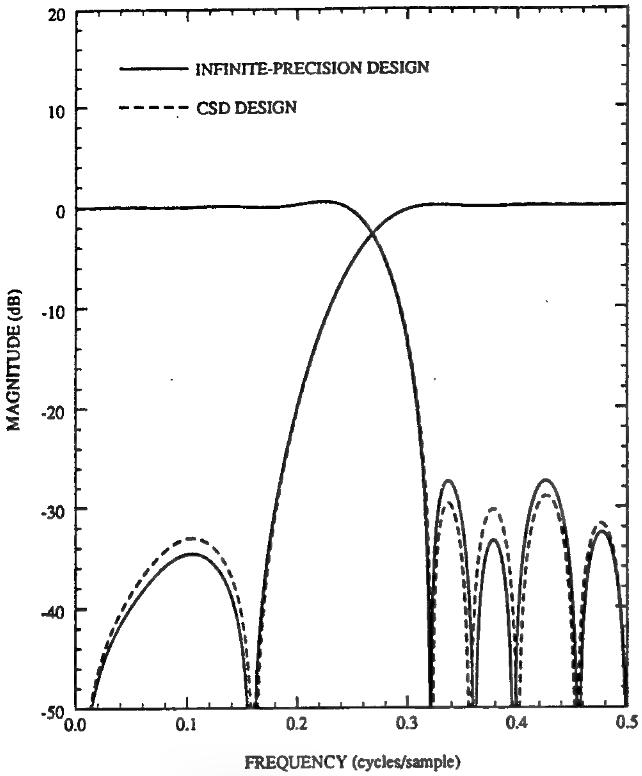


Fig. 2. Magnitude response plots of the CSD design and the infinite-precision design in Example 1.

TABLE I
CSD Coefficients For Example 1

n	$h_0(n)$	$h_1(n)$
0	$2^{-9} - 2^{-12}$	2^{-12}
1	$2^{-6} - 2^{-13}$	$-2^{-9} - 2^{-11} + 2^{-13} - 2^{-16}$
2	$2^{-6} + 2^{-10}$	2^{-9}
3	$2^{-6} + 2^{-9}$	$2^{-7} - 2^{-10} + 2^{-13} - 2^{-15}$
4	$-2^{-5} - 2^{-9}$	$-2^{-7} - 2^{-10} + 2^{-16}$
5	$-2^{-6} + 2^{-9}$	$-2^{-6} + 2^{-9} + 2^{-11} - 2^{-16}$
6	$2^{-4} + 2^{-13}$	$2^{-5} - 2^{-7} - 2^{-11} + 2^{-16}$
7	$-2^{-8} - 2^{-10}$	$2^{-5} - 2^{-9} + 2^{-11} - 2^{-14}$
8	$-2^{-3} + 2^{-8}$	$-2^{-4} + 2^{-8} - 2^{-10} - 2^{-12}$
9	$2^{-3} - 2^{-5}$	$-2^{-3} + 2^{-5} - 2^{-8} - 2^{-10}$
10	$2^{-1} - 2^{-7}$	$2^{-2} + 2^{-4} + 2^{-9} + 2^{-11} + 2^{-14}$
	SF $= 2^{0} + 2^{-1} - 2^{-4} + 2^{-7} - 2^{-10} - 2^{-14}2^{-14}$	

the scale factor, for implementing the entire analysis filter bank is 88.

IV. Application to Subband Image Coding

We wish to compare the performance of a multiplierless filter bank with other filter banks, when applied to subband image coding. The subband coding system used here was developed by Darragh and Baker [3], [4]. This system employed enumerative Laplacian quantization, which is made up of a scalar uniform threshold quantizer in cascade with an entropy encoder specifically tailored to the quantizer output statistics, for nonbaseband subbands, and differential pulse code modulation for baseband. Methods for allocating the rate among subbands predicated on subband quantizers were then established. The fixed-distortion subband coding algorithm (FDSBC) [3] solves the problem of minimizing the total bit rate subject to a constraint on allowable mean-square

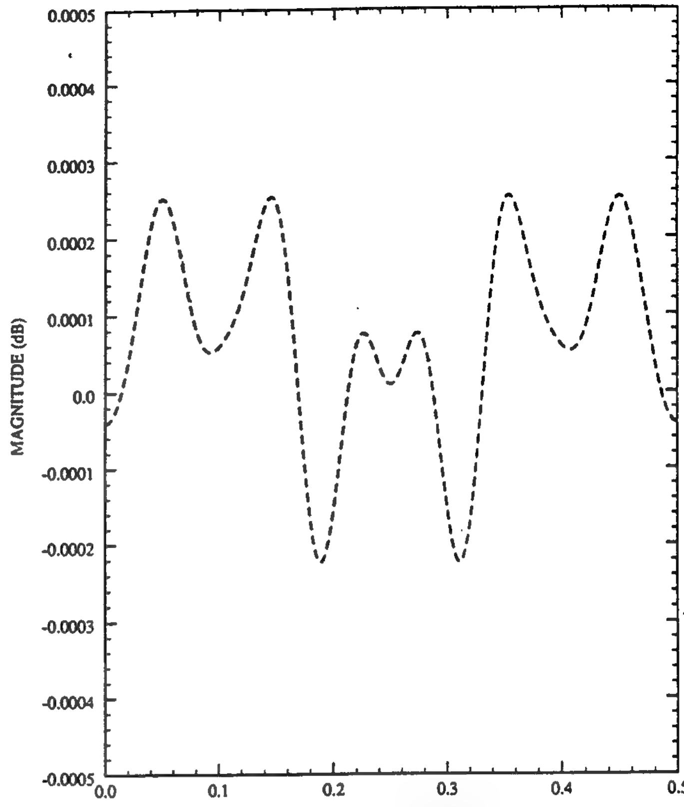


Fig. 3. System magnitude response plot of the CSD design in Example 1.

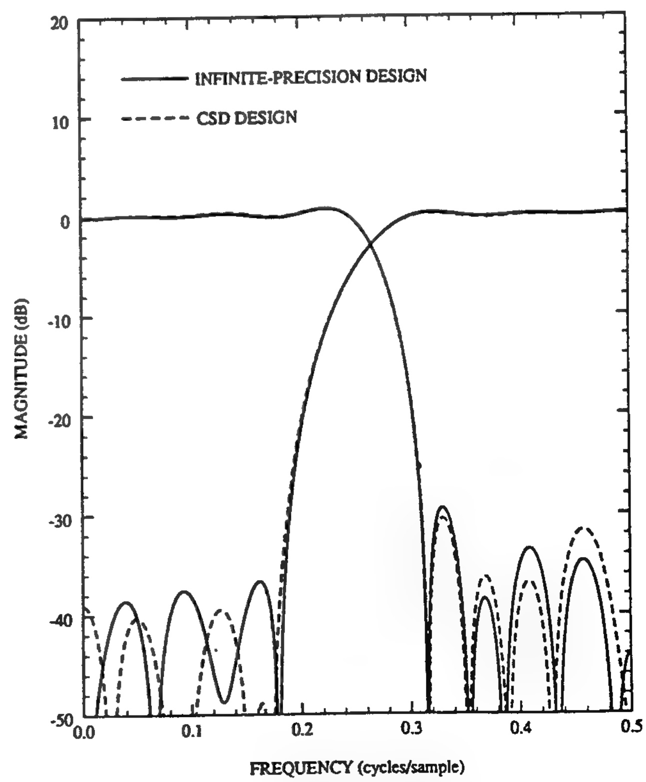


Fig. 4. Magnitude response plots of the CSD design and the infinite-precision design in Example 2.

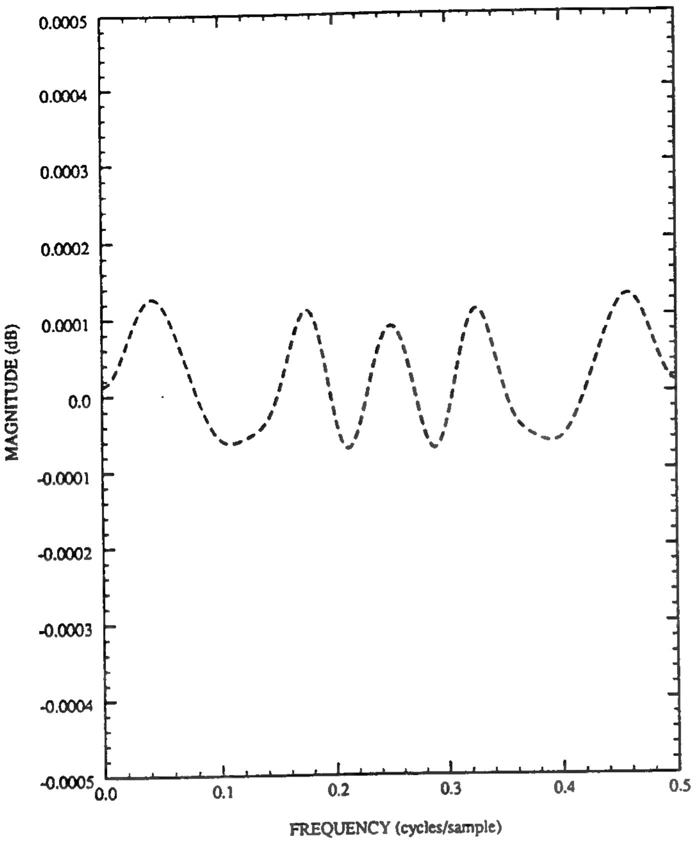


Fig. 5. System magnitude response plot of the CSD design in Example 2.

TABLE II
CSD COEFFICIENTS FOR EXAMPLE 2

n	$h_0(n)$	$h_1(n)$
0	2^{-12}	2^{-14}
1	$-2^{-6}-2^{-10}$	$-2^{-8}-2^{-11}-2^{-14}$
2	$2^{-8}+2^{-10}$	$2^{-9}-2^{-11}$
3	$2^{-5}-2^{-8}$	$-2^{-8}+2^{-10}-2^{-14}-2^{-16}$
4	$-2^{-6}+2^{-11}$	$-2^{-10}-2^{-13}-2^{-16}$
5	$-2^{-5}-2^{-7}$	$2^{-6}-2^{-9}+2^{-12}+2^{-16}$
6	$2^{-5}+2^{-8}$	-2^{-9}
7	$2^{-4}-2^{-6}$	$-2^{-5}-2^{-7}-2^{-12}$
8	$-2^{-3}+2^{-5}$	$2^{-7}-2^{-13}-2^{-15}$
9	$-2^{-4}+2^{-7}$	$2^{-4}+2^{-6}+2^{-8}+2^{-13}$
10	$2^{-2}+2^{-4}$	$-2^{-7}+2^{-10}-2^{-13}$
11	$2^{-1}+2^{-4}$	$-2^{-2}-2^{-4}+2^{-6}-2^{-11}-2^{-13}$
12		$2^{-1}-2^{-6}+2^{-8}-2^{-10}+2^{-16}$

$SF = 2^0 + 2^{-5} + 2^{-7} + 2^{-12}$

distortion, whereas the fixed-rate subband coding algorithm (FRSBC) [4] minimizes the mean-square error in the reconstructed image for a prescribed total bit rate. An original 256 × 256 pixel image, represented by 8bits/pixel, was encoded using the FDSBC algorithm, targeted at 33.36 dB, and the FRSBC algorithm, targeted at 0.5 bits/pixel, respectively. The filter banks tested are the 22-tap CSD filter bank (CSD-22) in Example 1, the 22-tap infinite-precision filter bank using the Lagrange–Newton method (Lagrange-22) in [6], and the well-known 32-tap quadrature mirror filter bank (QMF 32D), designated 32D in [7]. A two-level hierarchical structure, formed by the basic four-band equal-split structure as shown in Fig. 6, is used, yielding a total of 16 subbands. The rates, in bits per pixel (bpp), and the peak signal-to-noise ratios (PSNR's) are summarized in Table III. Here the PSNR

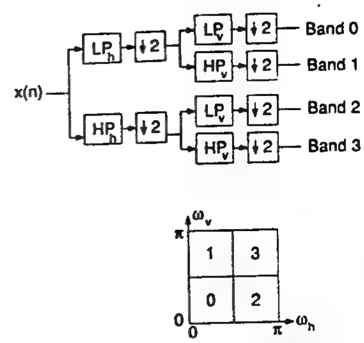


Fig. 6. The basic four-band equal-split analysis structure for subband image coding. LP_h: lowpass filtering in the horizontal direction. HP_h: highpass filtering in the horizontal direction. LP_v: lowpass filtering in the vertical direction. HP_v: highpass filtering in the vertical direction.

Fig. 7. Original image. (For color supplement see p. 392.)

Fig. 8. Reconstructed image using the CSD-22 filter bank and FDSBC algorithm, resulting in PSNR = 34.36 dB and bit rate of 0.807 bpp. (For color supplement see p. 392.)

TABLE III
PSNR AND RATE OF THE TESTED FILTER BANKS

Filter	PSNR (dB)		Rate (bpp)	
	FDSBC	FRSBC	FDSBC	FRSBC
CSD-22	34.36	30.96	0.807	0.447
Lagrange-22	34.39	30.98	0.808	0.448
QMF 32D	34.42	30.77	0.820	0.447

is related to mean square error d by

$$\mathrm{PSNR} = 10 \log_{10} \left[\frac{255^2}{d} \right] \mathrm{dB}.$$

Fig. 9. Reconstructed image using the Lagrange-22 filter bank and FDSBC algorithm, resulting in PSNR = 34.39 dB and bit rate of 0.808 bpp. (For color supplement see p. 392.)

Fig. 10. Reconstructed image using QMF 32D filter bank and FDSBC algorithm, resulting in PSNR = 34.42 dB and bit rate of 0.820 bpp. (For color supplement see p. 392.)

The original image and the reconstructed images using FDSBC for CSD-22, Lagrange-22, and QMF 32D are shown in Figs. 7, 8, 9 and 10, respectively. We notice that the bit rates, PSNR's and the subjective reconstructed image achieved by CSD-22 are almost the same as those of Lagrange-22. This confirms that the extremely small signal-reconstruction error resulting from CSD design is negligible in practice. Also, it is evident that the performance of CSD-22 is comparable to that of QMF 32D. The complexity of CSD-22, however, is much lower than the others. The Lagrange-22 filter bank requires 22 multipliers and 42 adders to implement the analysis filter bank; and the QMF 32D filter bank requires 16 multipliers and 32 adders/subtractors. Our CSD-22 filter bank, as described in Example 1, needs a total of only 85 adders/subtractors. The results here show that the proposed design technique provides an easy way to design low-complexity analysis/synthesis filter banks for high-performance subband image codecs.

V. Conclusions

A search technique has been presented for the design of multiplierless two-channel linear-phase FIR filter banks. The filter coefficients are represented by a CSD code, which makes it feasible to build the entire filter bank on a single chip using modern VLSI technology. Good filtering performance and nearly perfect signal reconstruction have been demonstrated through design examples. When applied to subband image coding the proposed design technique has yielded comparable coding performance and much less design complexity, compared with other infinite-precision design techniques. This feature of high performance with low design complexity should help make the recently popular subband image coding technique even more attractive.

References

[1] J. Woods and S. O'Neil, "Subband coding of images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 1278–1288, Oct. 1986.

[2] H. Gharavi and A. Tabatabai, "Sub-band coding of monochrome and color images," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 207–214, Feb. 1988.

[3] J. C. Darragh and R. L. Baker, "Fixed distortion subband coding of images for packet-switched networks," *IEEE J. Selected Areas in Communications*, vol. 7, pp. 789–800, June 1989.

[4] J. C. Darragh, "Subband and transform coding of images," Ph.D. dissertation, Electrical Engineering Department, University of California, Los Angeles, 1989.

[5] F. Lu and H. Samueli, "A 140-MHz CMOS bit-level pipelined multiplier accumulator using a new dynamic full-adder cell design," in *Proc. VLSI Circuits Symp.*, June 1990, pp. 123–124.

[6] B. R. Horng and A. N. Willson, Jr., "Lagrange multiplier approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, April 1990, pp. 1731–1734.

[7] J. D. Johnston, "A filter family designed for use in quadrature mirror filter banks," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, April 1980, pp. 291–294.

[8] T. Q. Nguyen and P. P. Vaidyanathan, "Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 676–690, May 1989.

[9] A. Avizienis, "Signed digit number representation for fast parallel arithmetic," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 389–400, Sept. 1961.

[10] M. Vetterli, "Multi-dimensional subband coding: some theory and algorithms," *Signal Processing*, vol. 6, pp. 97–112, April 1984.

[11] M. J. T. Smith and S. L. Eddins, "Analysis/synthesis techniques for subband image coding," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 1446–1456, Aug. 1990. I

[12] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[13] B. R. Horng, "New approaches to the design of two-channel FIR filter banks with application to subband image coding," Ph.D. dissertation, Electrical Engineering Department, University of California, Los Angeles, 1990.

[14] H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1044–1047, July 1989.

[15] D. Kodek and K. Steiglitz, "Comparison of optimal and local search methods for designing finite wordlength FIR digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-28, pp. 28–32, Jan. 1981.

[16] T. Lin and H. Samueli, "A 200-MHz CMOS $x/\sin(x)$ digital filter for compensating D/A converter frequency response distortion," *IEEE J. Solid-State Circuits*, vol. 26, pp. 1278–1285, Sept. 1991.

[17] R. Jain, P. Yang, H. Samueli, and T. Yoshino, "Architecture and floorplan design techniques for video-rate FIR filters," in *Proc. Int. Symp. Circuits Syst.*, May 1990, pp. 3027–3029.

Bor-Rong Horng (S'86–M'90) received the B.S. and M.S. degrees in power mechanical engineering from National Tsing-Hua University, Hsin-Chu, Taiwan, in 1980 and 1982, and the M.S.,Engr., and Ph.D. degrees in electrical engineering from the University of California, Los Angeles, in 1985, 1988, and 1990, respectively.

From 1982 to 1984 he was a Member of the Technical Staff at Aeronautical Research Laboratory, Taiwan. From 1985–1989 he was a Teaching Assistant/Associate, and from 1989–1990 he was a Research Associate, both in the Department of Electrical Engineering at UCLA. Since 1990 he has been with the Digital Communications Division of Rockwell International, Newport Beach, CA, where he is engaged in the design of telecommunication integrated circuits. His research interests include digital filter design, data compression, and integrated circuits for telecommunication applications.

Dr. Horng is a member of Sigma Xi.

Henry Samueli (S'75–M'81) was born in Buffalo, NY, on September 20, 1954. He received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of California, Los Angeles (UCLA) in 1975, 1976, and 1980, respectively.

From 1980 to 1985 he was with TRW, Inc., Redondo Beach, CA where he was a Section Manager in the Digital Processing Laboratory of the Electronics and Technology Division. His group was involved in the hardware design and development of military satellite and digital radio communication systems. From 1980–1985 he was also a part-time instructor in the Electrical Engineering Department at UCLA. In 1985 he joined UCLA full time where he is currently an Associate Professor in the Electrical Engineering Department. His research interests are in the areas of digital signal processing, digital filter design, analysis of finite wordlength effects in DSP systems, high-speed CMOS integrated circuit design, VLSI architectures for realizing DSP algorithms, and applications of VLSI technology to digital communication systems.

Alan N. Willson, Jr. (M'67–SM'73–F'78) was born in Baltimore, MD, on October 16, 1939. He received the B.E.E. degree from the Georgia Institute of Technology, Atlanta, GA, in 1961, and the M.S. and Ph.D. degrees from Syracuse University, Syracuse, NY, in 1965 and 1967, respectively.

From 1961–1964 he was with IBM, Poughkeepsie, NY. He was an instructor in Electrical Engineering at Syracuse University from 1965–1967. From 1967–1973 he was a Member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. Since 1973 he has been on the faculty of the University of California, Los Angeles, where he is now Professor of Engineering and Applied Science, in the Electrical Engineering Department. In addition, he has served the UCLA School of Engineering as Assistant Dean for Graduate Studies from 1977–1981 and is currently Associate Dean of Engineering. He has been engaged in research concerning computer-aided circuit analysis and design, the stability of distributed circuits, properties of nonlinear networks, theory of active circuits, digital signal processing, analog circuit fault diagnosis, and integrated circuits for signal processing. He is the editor of *Nonlinear Networks: Theory and Analysis* (New York: IEEE Press, 1974).

Dr. Willson is a member of Eta Kappa Nu, Sigma Xi, Tau Beta Pi, the Society for Industrial and Applied Mathematics, and the American Society for Engineering Education. From 1977–1979 he served as Editor of the IEEE Transactions on Circuits and Systems. In 1980 he was General Chairman of the Fourteenth Asilomar Conference on Circuits, Systems, and Computers. During 1984 he served as President of the IEEE Circuits and Systems Society. He is the recipient of the 1978 Guillemin–Cauer Award of the IEEE Circuits and Systems Society, the 1982 George Westinghouse Award of the American Society for Engineering Education, the 1982 Distinguished Faculty Award of the UCLA Engineering Alumni Association, the 1984 Myril B. Reed Best Paper Award of the Midwest Symposium on Circuits and Systems, and the 1985 W. R. G. Baker Award of the IEEE.

High-Speed Programmable FIR Prefilter Implementation

Linda T. Ying

Integrated Circuits and Systems Laboratory

Electrical Engineering Department

University of California, Los Angeles

Los Angeles, CA 90024.

Contact Author:

Linda T. Ying

Rockwell Telecommunications

M/S: 501-366

4311 Jamboree Rd.

Newport Beach, CA 92660.

E-mail: lying@nb.rockwell.com

Tel: 714-833-4923

Fax: 714-833-6211

High-Speed Programmable FIR Prefilter Implementation

by

Linda T. Ying

1 Abstract

For high-speed communications applications, most of the filtering requires narrow pass-band filters, which means that long FIR (Finite Impulse Response) digital filters are needed. It is well-known that one of the disadvantages of FIR filters is their high computational complexity. In order to reduce the number of adders and multipliers required, an attractive alternative for realizing the narrow band filters is to use a structure composed of a cascade of an RRS (Recursive Running Sum) prefilter and a corresponding magnitude response equalizer [1,2]. This report presents a silicon compiler for digital FIR RRS prefilter integrated circuits designed in the Mentor Graphics GDT CAD environment. The design goals, in decreasing order of importance, for this RRS prefilter are: high speed, small area, and low power dissipation. By using carry-save arithmetic in the hardware implementation, the critical path of the RRS prefilter is made independent of the data word length, which in turn means that the data word length does not affect the prefilter's speed. The critical path is composed of only two adders and a multiplexer. One noteworthy point is that the total number of adders required is independent of the prefilter order as a result of rewriting the transfer function. The prefilter is capable of implementing both the lowpass and highpass functions. Several prefilters can be cascaded in series to enhance the performance. A prototype chip has been generated from the compiler. It has been tested to be fully functional and it is expected to achieve a throughput-rate of about 175 MHz in a 1.2-μm CMOS process. The die size of the prototype chip is 4.0mm × 3.1mm (with pads).

2 Design Methodology

In order to fully realize the advantage of a prefilter/equalizer structure, the prefilter must be able to operate at the same high speed as the equalizer. While the equalizer FIR filter can be implemented simply with pipeline, the RRS prefilter has a recursive loop and requires a programmable delay line which makes its implementation more difficult than that of the equalizer. Since the equalizer is able to operate at 175 MHz, our target speed for the prefilter should be just as high, i.e. 175 MHz. Hence, full custom design datapath has to be used to meet the high speed requirement. In addition, power and area can also be optimized simultaneously. Therefore, a full custom design cell library is created in the Led layout editor tool for the leaf cells.

Top-down approach was used in our design. First, we decided the function and specifications of the chip and the best architecture to meet the requirements. We then investigated into each functional block and determined the leaf cells required. The leaf cells were manually laid out in Led. This allows better control of the critical delay path, area compaction and transistor sizing. After each cell was checked with the on-line GDT LRC (Layout Rule Checker) to make sure that it was free of design rule errors, its netlist was extracted and Lsim, a functional simulator, was used to check the cell's functional behavior. The cell was then optimized for timing with Hspice, a circuit simulator. With all the leaf cells ready, several Lx generators were written to produce the different functional blocks one at a time, with input parameters such as the input/output data word length, the width of the power supply/ground bus, and the maximum programmable delay value. These functional blocks were checked for design rule errors and functional behavior. Also, the critical delay path for the various blocks was simulated extensively to obtain a more accurate estimation of the worst-case propagation delay. Finally, the various blocks were assembled together with another Lx generator yielding the layout shown in Fig. 1. The resultant layout was checked for design rule errors using GDT LRC. Then, it was checked with another more thorough rule checker, Checkmate.

To verify that the connections of the various blocks are correct, and the resultant layout's functional performance is up to our expectation, Lsim was used to simulate the different cases. A Genie program was written to compare the Lsim simulation results with the expected results to further ensure the chip's functionality. In order to speed up the functional simulation, M-language functional models were written for some of the building blocks and leaf cells.

3 Introduction

This report presents the first integrated circuit prototype implementation of a high-speed programmable digital FIR prefilter. It is well known that one of the disadvantages of FIR digital filters is their high computational complexity. In order to reduce the number of adders and multipliers required, a structure using a cascade of a Recursive Running Sum (RRS) prefilter and a corresponding magnitude response equalizer has been proposed [1,2]. Other attractive prefilter schemes, such as prefilters based on the Dolph-Chebyshev function [3] and cyclotomic polynomials [4] have subsequently appeared. We have set high speed, small area, and low power dissipation as the design goals of our programmable prototype chip. The RRS structure was chosen for implementation. The IC was fabricated by MOSIS using 1.2-µm N-well technology. A photomicrograph of the prototype chip is shown in Fig. 1.

The basic structure of a lowpass RRS prefilter [1] with impulse response of length L is shown in Fig. 2. Its transfer function is:

$$H(z) = \frac{1}{L} \cdot \frac{1 - z^{-L}}{1 - z^{-1}} \tag{1}$$

Its implementation requires only two adders, (L+1) delay elements and a scaling multiplier. The number of adders used is independent of the prefilter order, which is an asset in creating a compact layout for a programmable structure. The frequency response of an RRS prefilter is the same as that of a length L rectangular time-domain window function. Therefore, the minimum stopband

attenuation that any RRS prefilter can provide is approximately 13dB. It would seem desirable to increase this rather modest level of stopband attenuation. In addition, the RRS prefilter's passband rolloff needs to be compensated. Hence, in order to simultaneously increase both the passband and stopband performance, the modified Simple Symmetric Sharpening (SSS) structure [2], as shown in Fig. 3, is of particular interest. It, however, requires one additional precise multiplier, as will be explained in Section 4.

4 Architecture

The factor limiting the speed (i.e., maximum data rate) of an RRS implementation is the time required for the computations performed in the recursive loop. The most commonly used methods to increase the maximum data rate for digital signal processing applications are word-level pipelining, retiming, and parallelism. However, none of these techniques can be carried out within a recursive loop as this would alter the filter's transfer function. Therefore, carry-save adders (CSAs) were used in our prototype chip to enhance its performance by pushing the carry propagation chain out of the recursive loop, thereby allowing the carry propagation to be performed with a pipelined adder. A straightforward implementation, shown in Fig. 4, gives the highest operating speed because the recursive loop is composed of only one CSA. However, two pipelined adders are required in this implementation, which consumes a substantial amount of area and imposes a considerable loading on the high-speed system clock. Hence, we decided to sacrifice a small amount of speed, and we implemented the structure shown in Fig. 5 which uses only one pipelined adder. The recursive loop is now composed of two CSAs.

To meet a greater variety of frequency response requirements, the prototype chip was designed with the capability of implementing both lowpass and highpass prefilters. Multiplexers are employed to specify whether or not to take the complement of the data in the recursive loop, thereby performing the simple lowpass-to-highpass transformation: $z \rightarrow -z$. As a result, the prefilter's speed is limited by two CSAs and a multiplexer, as shown in Fig. 6.

Since the two parallel branches within the shaded block of Fig. 3 should both be normalized by an identical factor, and since the RRS branch H_1 has an inherent dc gain of L_1, we must either: (1) use a precise programmable scaling multiplier of $1/L_1$ for H_1, or (2) scale up the data in the lower delay branch by a factor of L_1 and then perform a normalization at the output, after the addition of the two branches. Depending on the dynamic range requirements, the normalization in the second approach can be either a precise scaling or an approximate (power-of-two) scaling. The latter scheme is used in our design since a precise programmable *integer* multiplier of L_1 is easier to implement than a precise multiplier of $1/L_1$. Output normalization is then implemented with a barrel shifter, which is composed entirely of n-type pass-gates, which results in a compact layout.

5 Programmable Implementation

To allow our prototype chip to be programmable, several building blocks need to be programmable: the programmable delay line, the programmable integer multiplier L_1, and the programmable barrel shifter. Additional programmable features include the lowpass/highpass selection and a user-specified choice of implementing a stand-alone RRS prefilter or a modified SSS structure.

A DRAM using 3-T cells, shown in Fig. 7, is used to implement the programmable delay line (i.e., z^{-L} in Fig. 2). Since the DRAM block is being accessed serially, the address decoding scheme can be simplified by taking advantage of this characteristic. The reading of the first DRAM column is being done exactly L clock cycles after the writing of the same DRAM column. Therefore, a loadable counter is an ideal element for keeping track of the number of clock cycles that have evolved and initiating the read signal. After the read signal has been initiated, it can be propagated through the rest of the DRAM address columns. When it reaches the last DRAM column, it can be fed back to the first DRAM column and the whole cycle restarted again. In other words, the DRAM block is acting like a circular buffer. Since a whole DRAM column is accessed simultaneously whenever the column is being read or written, no row addressing is necessary. The column address

decoding circuits are simply a stage of C^2MOS shift registers, in contrast to a traditional address decoding implementation which would require a stage of address calculation circuits followed by a stage of address decoding circuits. Through the use of simplified address generation circuits, not only can area be saved, but the propagation delay time is also shortened.

There are three main operations for the DRAM block: precharge, read, and write. With high speed as a crucial design goal, separate read and write bit-lines are used–in other words, dual port DRAM cells are used–so that write can operate independently of read or precharge, sacrificing a rather small amount of area. In such cases, write will not be a constraint on the speed of the DRAM. The only timing constraint is that precharge and read should be non-overlapping to prevent a short-circuit current flow from power to ground, which would consume excessive power. Therefore, the maximum rate of operation of the DRAM is determined by the total time needed to precharge the bit-line and then to perform the read operation. Due to our high-speed requirements, a decimated clock with half the speed of the system clock is used with the DRAM. This effectively doubles the DRAM duty-cycle. The only drawback in using such a scheme is the need to use extra circuitry for demultiplexing the input bus and multiplexing the output bus. Since the clock used has been decimated by a factor of two, this automatically imposes a constraint that the programmable delay has to be even. Hence, a stage of multiplexer circuitry is needed to determine whether an extra latch stage needs to be bypassed, depending on whether L is odd or even. Interleaving had also been considered as an alternative to the decimated clock approach, but since it requires a complex clocking scheme, it did not seem to be the best approach for high-speed operation. The architectural block diagram of the DRAM is shown in Fig. 8.

6 Prototype Chip

A prototype chip which can function either as a stand-alone RRS prefilter or as the shaded part of Fig. 3 was fabricated through MOSIS using 1.2-μm HPCMOS34 technology. The selection of one of these two functions is achieved through the multiplexers shown in Fig. 9. If a single RRS

prefilter is needed, then the select signal is set to the appropriate value such that the multiplexers pick the branches that give the performance of a stand-alone prefilter. Using this structure, three of our prototype chips can be cascaded to implement the complete modified SSS structure of Fig. 3.

The prototype chip's datapath is shown in Fig. 10. In order to facilitate the chip's testing, a pseudo random number generator (PRNG) is included on-chip. It is a type II linear feedback shift register, designed using the algorithm outlined in [6]. The PRNG also serves as a buffer for the input data. A control signal is present to select whether the source of the input data should be from the input data bus or from the PRNG. The accumulator, as mentioned in Section 4, is composed of two CSAs and a multiplexer. The multiplier is implemented using the programmable canonic-signed-digit carry-save scheme described in [5]. Since the outputs of both the accumulator and the multiplier are in carry-save formats, a CSA block composed of two CSAs in series is necessary. This block converts the 4-bit vector to a 2-bit vector so that the resultant 2-bit vector can be fed into the vector-merge adder. The vector-merge adder is implemented with a six stage pipelined carry-ripple adder. The adders used in the implementation are transmission-gate adders [5]. A summary of the prototype chip is given in Table 1.

Table 1 Summary of the prototype chip

Technology	1.2-μm HPCMOS34 single poly double metal
Die size (with pads)	4.0mm × 3.1mm
Input word length	16 bits
Output word length	16 bits
Internal word length	23 bits
Number of pins	65
Maximum prefilter length	32
Testing results	fully functional
Yield	100% (25 parts fabricated, 25 parts fully functional)
Maximum data rate	175 MHz (simulated)

7 Testability and Testing Results

With over 35k input vectors tested on the LV500 tester, the chip has tested to be fully functional. The yield is an excellent 100% for the 25 parts fabricated. Since the target speed of the chip is about 175 MHz, in order to assist in the high-speed testing of the chip, a pseudo random number generator (PRNG) was placed on chip to achieve high fault coverage. The PRNG generates a white noise input. By connecting the prefilter chip with a D/A converter and then connect the results to a spectrum analyzer, the frequency response can be observed.

8 Conclusions

A silicon compiler for RRS and SSS digital FIR prefilter integrated circuits has been designed in the Mentor Graphics GDT CAD environment. The design goals for this prefilter are high speed, small area, and low power. By using carry-save arithmetic in the hardware implementation, the critical path of the RRS prefilter is made independent of the data word length, which in turn means that the data word length does not affect the speed of the prefilter. To be precise, the critical path is composed of two CSAs and one multiplexer. Three of our prefilter ICs can be cascaded to enhance performance and implement the complete SSS prefilter of Fig. 3. A prototype chip has been generated from the Lx-language compilers and it is tested to be fully functional with over 35k input vectors. The yield is 100% for the 25 parts fabricated. It is expected to achieve a throughput-rate of 175 MHz (simulated) in a 1.2-µm CMOS process. The die size of our prototype chip is 4.0mm × 3.1mm (with pads).

References

[1] J. W. Adams and A. N. Willson, Jr., "A new approach to FIR digital filters with fewer multipliers and reduced sensitivity," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 277-283, May 1983.

[2] J. W. Adams and A. N. Willson, Jr., "Some efficient digital prefilter structures," *IEEE Trans. Circuits Syst.*, vol. CAS-31, pp.260-266, Mar. 1984.

[3] P. P. Vaidyanathan and G. Beitman, "On prefilters for digital FIR filter design," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 494-499, May 1985.

[4] R. J. Hartnett and G. F. Boudreaux-Bartels, "On the use of cyclotomic polynomial prefilters for efficient FIR filter design," *IEEE Trans. Signal Processing*, vol. 41, pp. 1766-1779, May 1993.

[5] K. Y. Khoo, A. Kwentus, and A. N. Willson, Jr., "An efficient 175MHz programmable FIR digital filter," in *Proc. 1993 IEEE Int. Symp. Circuits Syst.*, May 1993, pp. 72-75.

[6] M. Abramovici, M.A. Breuer, and A.D. Friedman, *Digital Systems Testing and Testable Design*. New York: Computer Science Press, 1990, pp. 432-441.

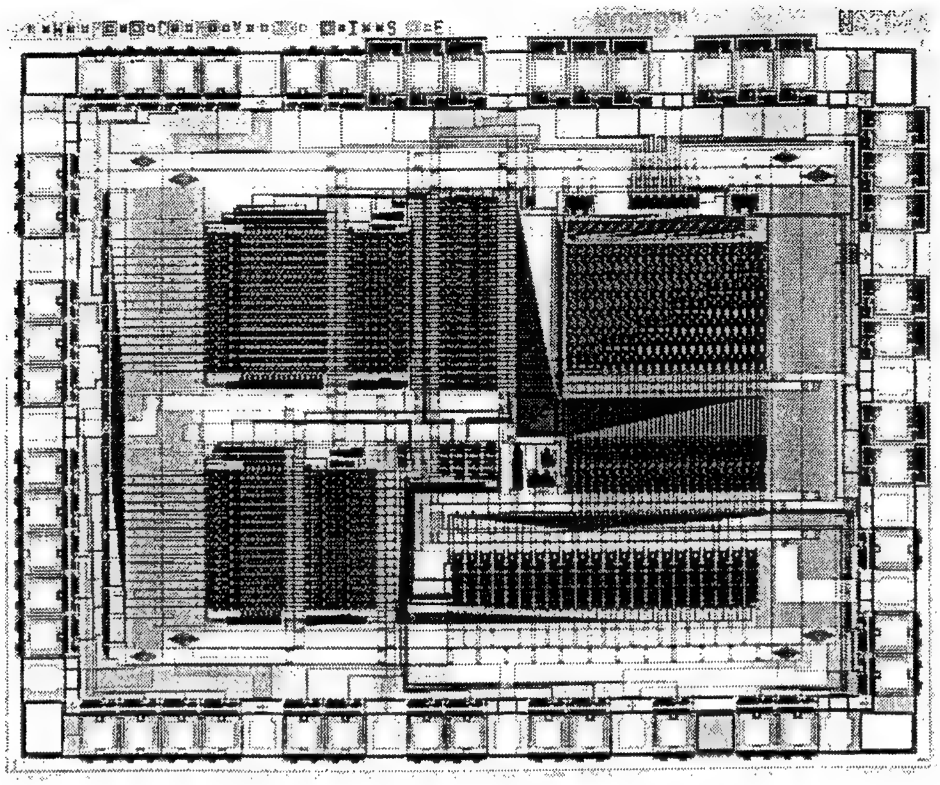


Fig. 1. Prototype chip photomicrograph.

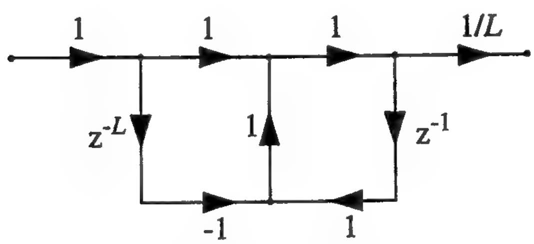


Fig. 2. RRS prefilter realization.

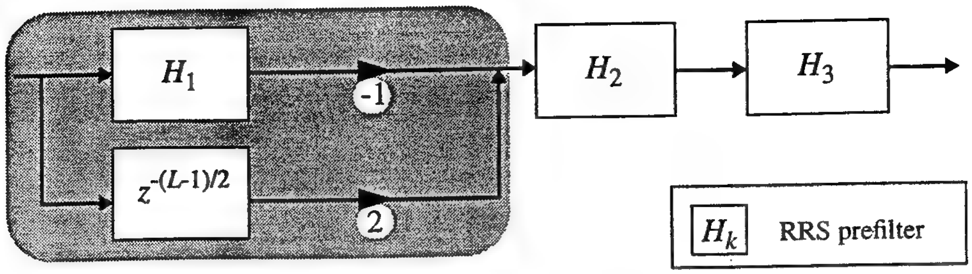


Fig. 3. Modified SSS structure.

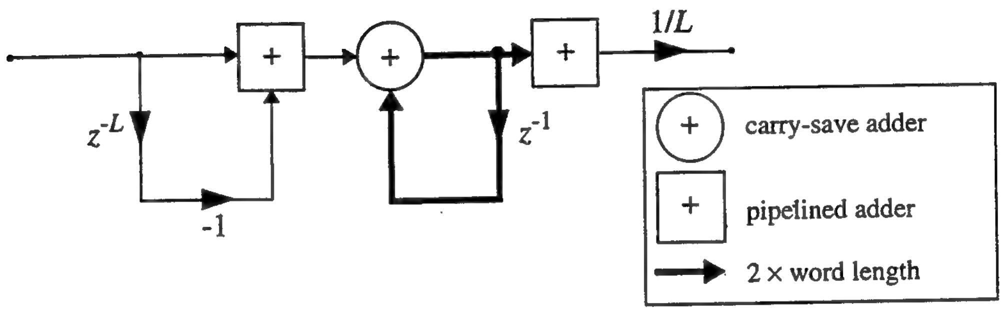


Fig. 4. Implementation with two pipelined adders and a carry-save adder.

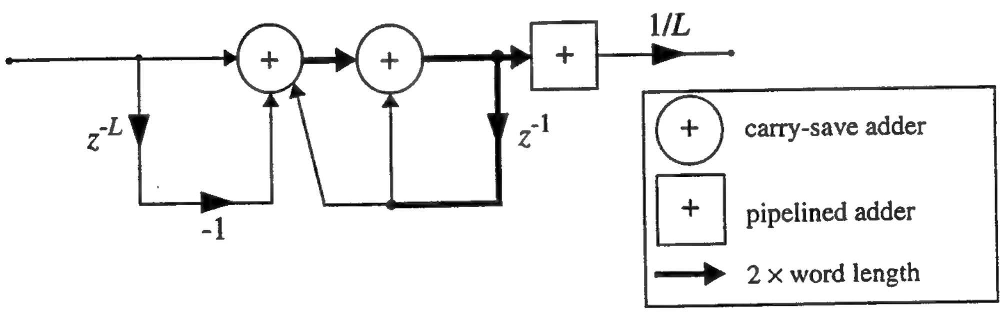


Fig. 5. Implementation with a pipelined adder and two carry-save adders.

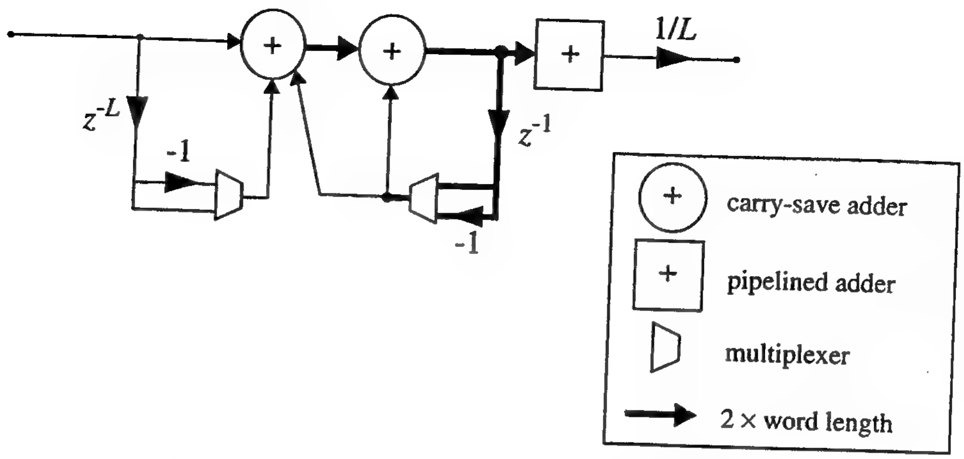


Fig. 6. Implementation of a lowpass/highpass RRS prefilter.

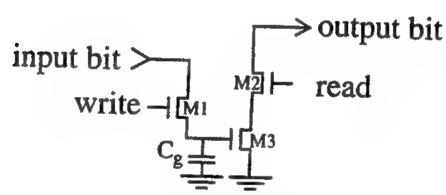


Fig. 7. A 3-T DRAM cell.

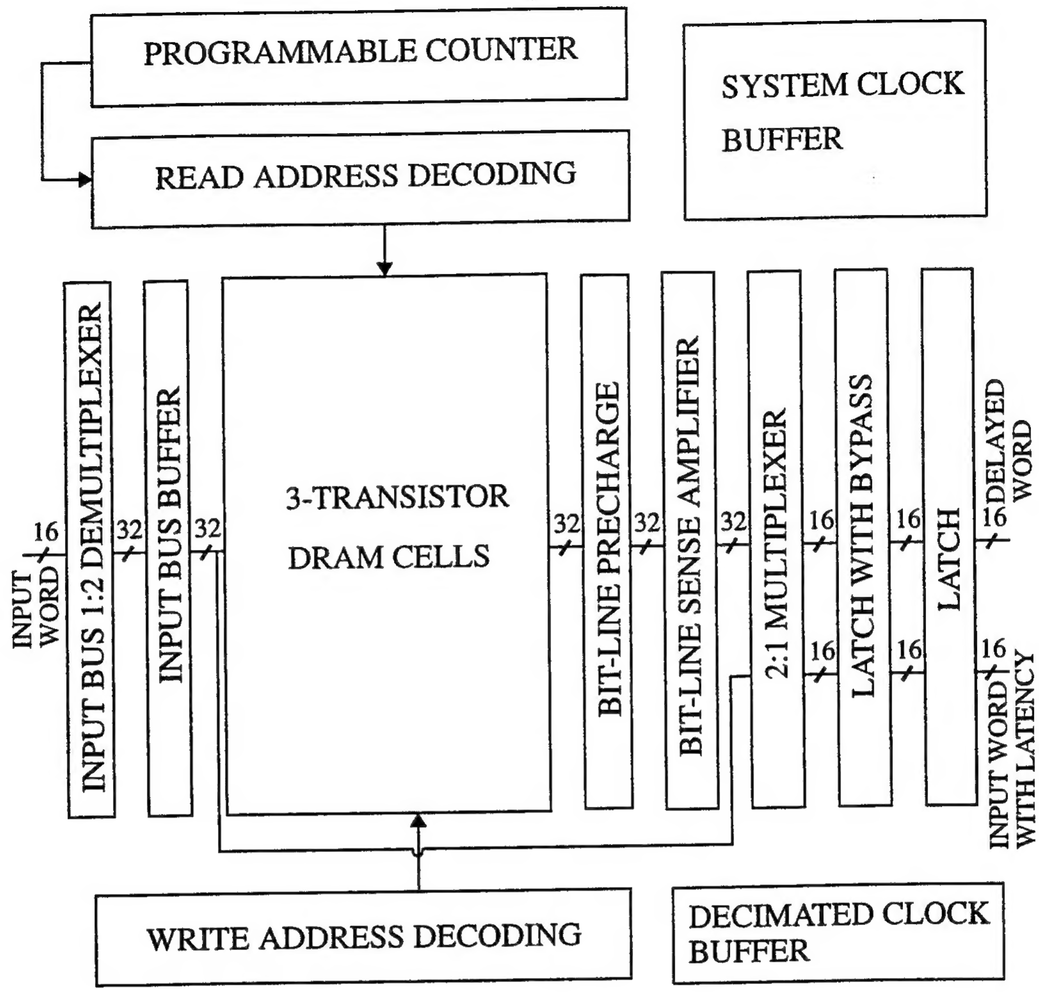


Fig. 8. Architectural block diagram of the DRAM.

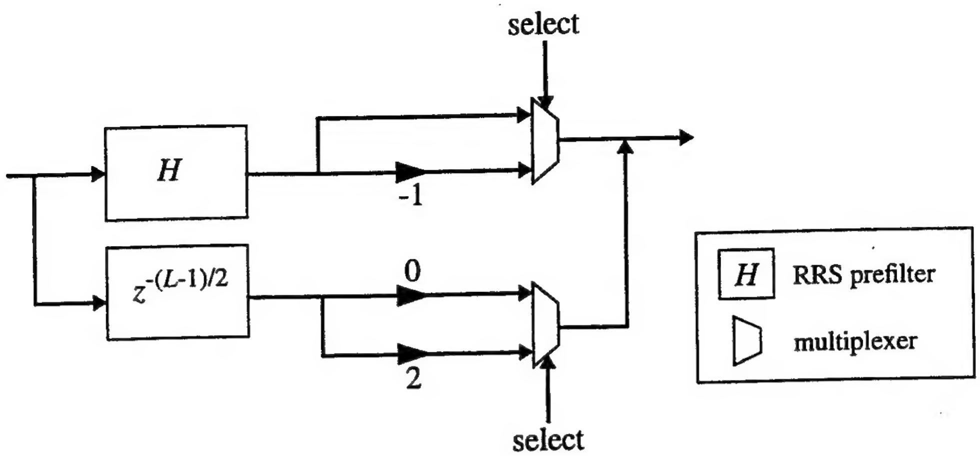


Fig. 9. Abbreviated block diagram of the prototype chip.

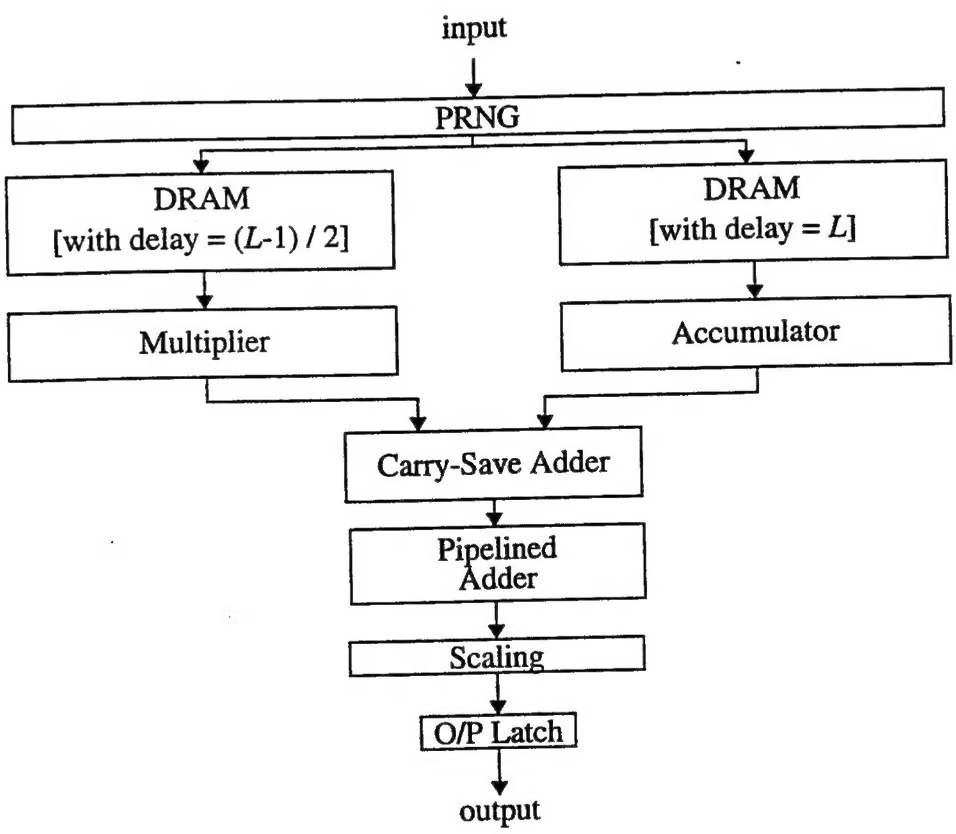


Fig. 10. Datapath of the prototype chip.

REPORT OUTLINE

1. Title page.
2. Abstract.
3. System Overview. Include motivation for designing the chip. Is VLSI appropriate? Design methodology and why this methodology was chosen. Does this design satisfy the system requirements? What is unique about this project? What novel ideas or elegant solutions does the design include? Provide list of resources used, including software analysis.
4. Implementation and engineering considerations (bulk of the report):
   - Specifications: functional, timing, electrical and environmental.
   - Tradeoffs: architectural and circuit tradeoffs, floorplanning and interconnect approaches, and I/O considerations. Place emphasis on why you did what you did.
   - Timing and Critical Paths. What clocking scheme is used and why? Which paths are critical? Have you simulated or measured their delays?
   - Block Diagram, Logic/Circuit Diagrams, and Algorithms.
   - Final Layout Plot (annotate so various blocks can be identified).
   - Verification and Simulation: how did you assure that the chip would work as specified.
   - Testing: how did you, or will you, test this part with I/O pins only? What test equipment did you use? Actual test results, if available, should be summarized.
   - Chip statistics: die size, total power, number of transistors, density of layout, maximum clock speed, etc.
5. Summary
6. References

JUDGING CRITERIA

Entries will be judged based on the following criteria:

30% - Soundness of engineering, including evaluation of trade-offs and ingenuity at all levels of design activity.

30% - Clarity of functional specification, system description, and chip description.

30% - Producibility and testability of the chip including explanations of any special test procedures you have included to make the part producible.

10% - Design methodology selected and explanation of why that methodology was chosen.

PRIZE AWARDS

Over $50,000 in cash and equipment (Texas Instruments Laptop Computers and a Sun workstation) will be awarded to the winning students and schools of the winning students.

First, second and third prizes will be awarded in each category and class.

One design entry will be selected as the overall grand prize winner.

1994 STUDENT VLSI DESIGN CONTEST

SPONSORED BY

Mentor Graphics®

ELECTRONIC DESIGN

HEWLETT PACKARD

Sun

TEXAS INSTRUMENTS

1994 STUDENT VLSI DESIGN CONTEST

Over $50,000 in cash and equipment will be awarded

ENTRY DEADLINE: JUNE 15, 1994

This contest is designed to promote excellence in education for integrated circuit designers at colleges and universities. It provides competition between students participating in classes and in research who design and fabricate integrated circuits.

The contest has two categories - digital design and analog/mixed signal design. Each category will have two classes - novice and experienced. The novice class is intended for students taking an introductory quarter/semester class in which they design an integrated circuit. The experienced class is intended for students that have already completed the introductory class and are continuing in an advanced course or doing research that involves the design and fabrication of integrated circuits.

Both contest categories are for the most innovative IC design project created by a student or team of students who are or have been enrolled in VLSI classes at schools participating in Mentor Graphics' North American Higher Education Program. Entries are due June 15, 1994. A panel of judges from the contest sponsors will evaluate the entries and determine the winning designs. Contest winners will be announced in August, 1994.

CONTEST CRITERIA

The contest is open to any full-time undergraduate or graduate student at colleges and universities participating in Mentor Graphics' North American Higher Education Program. Students are eligible to enter provided they have not, before completion of their projects, been employed as integrated circuit designers in industry or in full-time IC design positions in their respective universities.

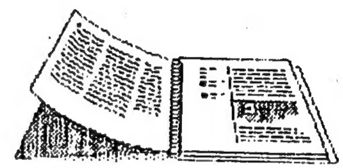
To be eligible for the contest a chip must meet functional needs and be non-proprietary. It is desirable, but not mandatory, that the chips be fabricated and tested. If your chip is tested, you must report the results even if they were negative. A testing procedure should be included whether or not you have tested the chips.

All work on the project must be done using facilities and equipment provided by the university. The following Mentor Graphics IC design software must be used for the layout of your design: GDT Designer or IC Station for digital designs; and IC Station for analog/mixed signal designs. The design style can be full-custom or standard-cell.

REPORT SPECIFICATIONS AND HINTS

Judging will be based on a written report that must conform to the outline described below.

Each report shall be a total of 12 pages or less in the novice category and 18 pages or less in the experienced category. Page count includes the title page, text, figures, etc. No pages beyond this will be accepted. Text should be 10 pt and double spaced. The pages must be 8.5 x 11 inches, except one (optional) for a layout that may fold out as large as a C size sheet. Each paper is expected to follow the form specified below, including all items appropriate for the specific project.



Since projects are judged solely upon written reports, clarity and completeness of the report are of paramount importance. Writing style should be clear and concise. Remember that the judges' expertise may not be in the area of your project. Make your explanations straightforward and understandable.

Design for testability is an important issue. Discuss testing issues you have considered in the design and approaches you took or will take in testing. Engineering specifications and performance statistics can be efficiently presented in tabular form.